НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Web-редактор сценарію гідроакустичного експерименту»

Виконав:
студент IV курсу, групи ТР-61
Сурін Богдан Сергійович _____

Керівник:
Старший викладач,
Колумбет Вадим Петрович _____

Консультант з постановки задачі побудови
веб інтерфейсу гідроакустичного експерименту
Старший викладач,
Варава Іван Андрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

1 Сурін Богдан Сергійович_____
(прізвище, ім’я, по батькові)

1. Тема роботи «Web-редактор сценарію гідроакустичного експерименту» ___
_____
керівник роботи Колумбет Вадим Петрович старший викладач _____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _платформа Microsoft Visual Studio 19, мова програмування C# для клієнтської сторони та мова програмування Node.js для серверної_____
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розробити веб-додаток для задання початкових умов гідроакустичного експерименту та реалізувати код Web-сервісу. Створити зручний користувацький інтерфейс для вводу початкових даних та отримання результату у зручному форматі. Візуально зобразити морські об’єкти та згенерувати файл XML розширення що буде містити дані об’єктів та параметри солоності для подальшої обробки у системі. _____ 5. Перелік ілюстративного матеріалу
«Постановка задачі», «Сфери застосування даного рішення», «Існуючі рішення», «Використані технології», «Архітектура системи», «Головний екран Web-сервісу», «Головні елементи інтерфейсу», «Введення нових направлень», «Алгоритм генерації

сигналу», «Кінцевий результат», «Необхідно для користування», «Висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі побудови веб інтерфейсу гідроакустичного експерименту	Варава І.А. старший викладач	16.10.2019	07.02.2020

7. Дата видачі завдання ”__”_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	14.10.2019	
2.	Вивчення та аналіз задачі	14.10.2019-23.12.2020	
3.	Розробка архітектури та загальної структури системи	02.02.2020-03.03.2020	
4.	Розробка структур окремих підсистем	04.03.2020-14.04.2020	
5.	Програмна реалізація системи	15.04.2020-19.05.2020	
6.	Оформлення пояснювальної записки	20.05.2020-05.06.2020	
7.	Захист програмного продукту	18.06.2020	
8.	Передзахист	10.06.20	
9.	Захист	18.06.20	

Студент _____ Сурін Б.С._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Колумбет В.П._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було створення веб редактору сценарію гідроакустичного експерименту. Було виконано огляд існуючих програмних застосунків для редагування та моделювання гідроакустичного експерименту, виявлено їх переваги та недоліки.

Користувацький інтерфейс програми дозволяє ввести вхідні дані, необхідні для подальших розрахунків, серед яких початкові та кінцеві координати об’єкта, його кут та швидкість, амплітуда хвиль, глибина. Частота та профілі солоності моря. Після візуалізації та занесення даних до відповідних таблиць, додаток експортує наведені вище дані з таблиць у форматі json виводить графіки віддаленості морського об’єкта від гідрофона у кожен момент часу, хвильову картину та генерує кінцевий сигнал для будь-якої кількості об’єктів.

Записка містить 50 сторінок, 23 рисунків та 20 посилань.

ABSTRACT

The aim of the work was to create a web editor for a sonar search experiment. An overview of existing software applications for editing and modeling a sonar experiment was performed, their advantages and disadvantages were identified.

The user interface of the program allows you to enter the input data needed for further calculations, including the initial and final coordinates of the object, its angle and speed, wave amplitude, depth. Frequency and salinity profiles of the sea. After visualizing and entering data into the appropriate tables, the application exports the above data from tables in json format displays graphs of the distance of the marine object from the hydrophone at any time, a wave picture and generates a final signal for any number of objects.

The note contains 50 pages, 23 figures and 20 references.

ЗМІСТ

Перелік умовних позначень 7
Вступ 8
1.Задача побудови веб інтерфейсу гідроакустичного експерименту 10
1.1. Розробка клієнтської частини програмного засобу 10
1.2. Природа звуку 10
1.3. Розподіл сферичних хвиль 14
2. Аналіз проблеми існуючих систем 17
2.1. Коливальний процес системи з розподіленими сталими 17
2.2. Опір плоских та сферичних хвиль 18
3. Засоби розробки 20
3.1. Середовище розробки Visual Studio 20
3.2. Мова програмування C# 21
3.3. Об'єктно-рольове моделювання 22
3.4. Платформа NET Framework 24
3.5. Технологія для розробки інтерфейсу 27
3.6. Середовище Microsoft SQL Server 28
4. Опис програмної реалізації 30
4.1. Архітектура бази даних 32
4.2. Технологія для розробки програмного продукту 32
4.2.1 Впровадження залежностей Dependency injection 32
4.2.2 Стиль CQRS 32
4.2.3 Архітектура MVC 33
4.2.1 Unit тести 35
4.2. Концептуальна модель 36
4.3. Інтерфейс користувача 38
4.4. Структура таблиць бази даних 38
5. Робота з програмною системою 42
5.1. Системні вимоги та інсталяція 42

5.2. Сценарій роботи з програмним продуктом 42
Висновки 47
Список використаних джерел 49
Додаток А 51
Додаток Б 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

● ГАС — апаратний комплекс, гідроакустична станція, яка використовує явище розповсюдження гідроакустичних хвиль в морях, океанах та інших водних середовищах. Ці станції призначені для вирішення задач навігації, зв’язку, спостереження;

●CLR— Common Language Runtime

●CIL—єдиний проміжний байт-код Common Intermediate Language

● FFT — Fast furrier transform

● MVC—Model view controller

● ОС — операційна система

● ПЗ — програмне забезпечення

● IDE — Integrated Drive Electronics

ВСТУП

Гідроакустика - наука про явища, що відбуваються у водному середовищі і пов'язаних з поширення, випромінюванням і прийомом акустичних хвиль. Це порівняно молода наука, що швидко розвивається в даний час і має, безсумнівно, велике майбутнє. Її появі передував довгий шлях розвитку теоретичної та прикладної акустики. Дана галузь науки включає питання розробки і створення гідроакустичних засобів, призначених для використання у водному середовищі. В даний час ведуться дослідження з використання низьких частот в гідроакустичних засобах, а також дослідження далекого поширення звуку в глибокому океані. Широке застосування знаходять методи математичного моделювання умов поширення звуку на цифрових і аналогових електронно-обчислювальних машинах з подальшим використанням результатів моделювання для вироблення відповідних вимог до гідроакустичної апаратури.

Метою гідроакустичного експерименту є збір даних на основі звукових хвиль. Вони виступають єдиним видом сигналів, які можуть розповсюджуватися на значні відстані з малим затуханням в океанах і морях.

Актуальність даної теми полягає в тому, що з появою нових технологій та зі збільшенням кількості даних для подальшої обробки з’являється потреба в більш гнучких методах опрацювання інформації.

Ефективність роботи будь-якого додатку з моделювання, де уся інформація залежить від ряду факторів, одним з яких є легкий доступ до взаємодії з даними. А як відомо, одним з найважливіших умов успішного функціонування комунікацій між студентом та викладачем є ефективний обмін інформацією. Проте, часто, це затратний процес. Наприклад, розглянемо налаштування морських об’єктів у просторі, де потрібно затвердити їх координати, швидкість в певний проміжок часу, а також стан водойм, глибини та профілі солоності. Щоб зручно працювати з такою кількістю даних без використання великих об’ємів паперу, виникає необхідність автоматизувати цей процес, таким чином вдосконалити його. Саме тому, розробка автоматизованої системи для роботи з масивом даних гідроакустичного експерименту є актуальною.

А для контролю даних в системі потрібен кваліфікований працівник, роль якого виконує додатковий персонал.

Тому метою є розробка веб-інтерфейсу, з необхідним функціоналом для моделювання початкових умов гідроакустичного експерименту з можливістю графічного відображення об’єктів у віртуальному просторі. Даний інтерфейс зможе полегшити обробку великого масиву даних на початкових етапах моделювання.

Програмний додаток розроблявся на мові програмування C#, платформи .NET Framework.

1. ЗАДАЧА ПОБУДОВИ ВЕБ ІНТЕРФЕЙСУ ГІДРОАКУСТИЧНОГО ЕКСПЕРИМЕНТУ

1.1 Розробка клієнтської частини програмного засобу

Метою розробки є створення веб інтерфейсу, у якому користувач потребує значно меншої кількості дій на етапі задання та відображення початкових умов. Збільшити зрозумілість і зручність користування шляхом
побудови інтуїтивно зрозумілого користувацького інтерфейсу.
Призначенням даного веб застосунку є надання інтуїтивного інтерфейсу для визначення початкових даних, їхнього задання у графічному інтерфейсі та їх подальший експорт для наступної обробки.

Програмний додаток розробляється для персональних комп’ютерів на базі операційної системи Windows 10.
Вхідні дані записуються користувачем в спеціальному вікні додатку.
Вихідні дані це введені початкові умови, які відображаються у візуальному вигляді.
Необхідні можливості, які надає веб додаток, є:
— ввід початкової інформації експерименту з комп’ютера користувача;
— можливість перегляду введених об’єктів у графічному інтерфейсі;
— збереження, редагування та видалення інформації до бази даних;

1.2 Природа звуку

Різноманіття фізичних явищ включає в себе велику категорію коливальних рухів, що підкоряються загальним закономірностям незважаючи на те, що природа їх різна. Це можуть бути механічні, електромеханічні, електромагнітні або термодинамічні коливання. Коливання - це зміна стану, що характеризуються тим або іншим ступенем повторюваності в часі.

Дослідження коливальних процесів має дуже важливе значення для сучасної науки і техніки, особливо для таких галузей, як приладобудування, радіотехніка, технічна акустика або суднобудування. Але в першу чергу нас цікавлять механічні коливання, так як саме з ними доводиться мати справу при роботі джерел акустичних хвиль у воді та повітрі. До механічних коливань слід віднести коливання корпусу корабля, балок будівель і споруд, явища припливів і відливів, коливання щільності і тиску частинок середовища, в якій працює джерело звуку. Залежно від закону, за яким повторюється рух, коливальні процеси можуть бути періодичними або неперіодичних. Періодичним коливальним процесом називається такий процес, при якому стан тіла, взятого в деякий момент часу, який характеризується тією чи іншою фізичною величиною, знову повторюється через певний проміжок часу *T*, званий періодом коливань. Для математичного опису періодичних коливальних процесів використовуються періодичні функції. Значення фізичних функцій, взятих через даний інтервал, що дорівнює періоду, збігаються. Це виражається рівністю

$$f(t+T) = f(t)$$

Як приклад періодичних коливань можна назвати рух математичного маятника при відхиленні на малий кут і відсутності сил тертя. Такий рух маятника відбувається за гармонійним законом (Рисунок 1.1).

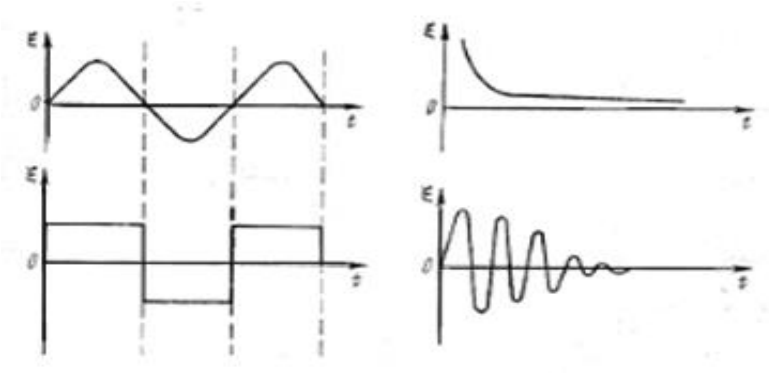

Рисунок 1.1- періодичні та неперіодичні коливання

Неперіодичні коливальні процеси не можуть бути описані періодичними функціями. Неперіодичний процес має період, який прямує до нескінченності. Прикладом неперіодичного коливального процесу може бути загасаючий коливальний процес, де

тіло, виведене з положення рівноваги, швидко прагне повернутися до свого початкового стану.

Тіло, що коливається, яке ми будемо називати коливальною системою, може віддавати свою енергію в зовнішнє середовище, стаючи джерелом звуку. Ця енергія передається на відстань, так як частинки (молекули) середовища є мініатюрними коливальні системи, пов'язані один з одним пружними зв'язками: якщо одна з частинок виведена з положення рівноваги, то сили, що діють на неї з боку сусідніх частинок, змусять її знову повернутися до стійкого положення. Згідно із законом рівності дії і протидії сусідні частинки також підпадають під вплив зміщуючих сил і виводяться з положення рівноваги. Однак через свою інерцію вони повторюють рух попередніх частинок з деяким запізненням. Таким чином, зміна, яка виникла в певній ділянці середовища, поступово поширюється, захоплюючи частинки все далі віддалені від джерела звуку. Такий рух називається хвильовим. Частинки середовища групуються, створюючи області згущення, і розходяться, створюючи області розрідження. Процес поширення станів згущення і розрідження від однієї ділянки або шару середовища до іншого називається акустичної хвилею. Швидкість, з якою поширюється акустична хвиля, називається швидкістю звуку c. Відстань між двома сусідніми шарами, що знаходяться в однаковому стані (згущення або розрідження), називається довжиною хвилі λ. За час одного повного коливання джерела акустичних хвиль коливальний процес поширюється на відстань, яка дорівнює довжині хвилі. Якщо протягом однієї секунди відбудеться f повних коливань, то хвилі розповсюджуватися на відстань

$$\lambda f = c$$

Слід зазначити, що частинки середовища не захоплюються рухається акустичної хвилею, а здійснюють коливальні рухи біля свого положення рівноваги. У цьому легко переконатися, спостерігаючи за пробкою, яка підстрибує на хвилях води. Пробка буде опускатися і підніматися, тобто здійснювати коливальні рухи, і разом з цим буде зберігати своє місцеположення. Швидкість поширення акустичної хвилі і швидкість руху частинок середовища - різні поняття. Швидкість, з якою коливається частка щодо свого положення рівноваги, називається коливальної швидкістю. Вид

хвиль, що поширюються в середовищі, залежить від пружних властивостей середовища.

Для з'ясування характеру цієї залежності розділимо подумки середу на ряд тонких, дотичних один з одним шарів, перпендикулярних до напрямку поширення хвиль. Рідини і гази практично не надають пружного опору зрушенню шарів. При спробі зблизити два сусідніх шару або видалити їх один від одного виникають повертають сили, що перешкоджають деформації стиснення і розтягування.

Коливання часток відбуваються в напрямку сил які повертають. Тому в середовищі де відсутні сили пружності, можливі лише такі хвилі, в яких коливання частинок збігаються з напрямом поширення хвилі. Такі хвилі називаються поздовжніми, так як частинки середовища рухаються уздовж хвилі. Прикладом поздовжніх хвиль є акустичні хвилі в воді і повітрі. Поздовжні хвилі поширюються і в твердих тілах, проте тут поряд з ними поширюються і поперечні хвилі. Поперечні хвилі в твердому тілі викликаються деформацією зсуву і поширюються в напрямку, перпендикулярному до напрямку зміщення частинок. У акустичної хвилі, що розповсюджується в пружною середовищі (наприклад, в повітрі), стиснення і розрідження відбуваються так швидко, що обмін теплом при цьому не встигає відбутися.

Стиснення середовища без відводу тепла носить назву адіабатичного стиснення. Для визначення швидкості поширення акустичної хвилі малої амплітуди можна користуватися тією ж формулою, що і для швидкості окремого слабкого імпульсу стиснення:

$$c_0 = \sqrt{\gamma \frac{p_0}{\rho_0}}$$

де p_0 - статичний тиск на рівні моря при 0^oC: ρ_0=101396,2 $H/м^2$; ρ_0 - статична (початкова) щільність: ρ_0=1,29 $кг/м^3$; γ - відношення теплоємності при постійному тиску і постійному обсязі. Для повітря γ = 1,4, для води γ = 1.

1.3 Розподіл сферичних хвиль

Припустимо, що в необмеженому середовищі знаходиться сфера, радіус якої малий у порівнянні з довжиною хвилі. Якщо радіус сфери буде періодично відхилятися від свого середнього положення, то навколо неї утворюються сферичні хвилі. Хвильові поверхні представлять собою концентричні сфери, і площа фронту хвилі буде $S = 4\pi r^2$ (Рисунок 1.2)

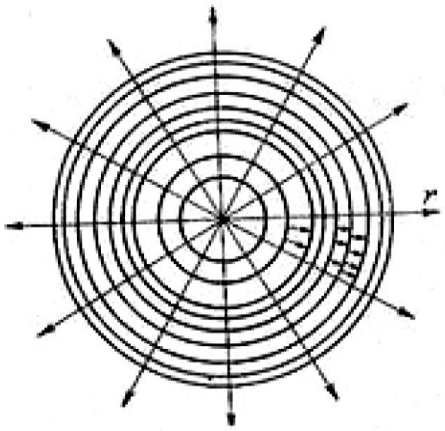

Рисунок 1.2 -фронт сферичної хвилі

Акустичне поле в даному випадку має повну симетрією щодо центру сфери, і стан середовища в будь-якій точці буде визначатися відстанню *r* від джерела і часом *t*. Хвильове рівняння сферичних хвиль має вигляд

$$\frac{\partial^2 (rp)}{\partial t^2} = c^2 \frac{(rp)}{\partial r^2}$$

Загальне рішення хвильового рівняння являє собою сукупність двох сферичних хвиль: хвилі, що розходиться від джерела зі швидкістю *c*, і хвилі, що сходиться до джерела з тією ж швидкістю.

$$p(rt) = \frac{1}{r} f_1\left(t - \frac{r}{c}\right) + \frac{1}{r} f_2\left(t + \frac{r}{c}\right)$$

член $\frac{1}{r} f_1\left(t - \frac{r}{c}\right)$ це сферична хвиля тиску, яка розходиться в різні боки від джерела, а член $\frac{1}{r} f_2\left(t + \frac{r}{c}\right)$ зворотна сферична хвиля тиску, що сходиться до джерела.

Сферичні хвилі, що сходяться до джерела, можуть виникнути, наприклад, при відображенні від поверхонь, що мають сферичну форму. У більшості випадків доводиться мати справу тільки з розбіжними сферичними хвилями.

Приватне рішення хвильового рівняння для прямої сферичної хвилі має вигляд

$$p(rt) = \frac{1}{r} f_1\left(t - \frac{r}{c}\right)$$

Рішення хвильового рівняння для випадку розходяться гармонійних сферичних хвиль може бути представлено в комплексній формі виразом

$$p = \frac{1}{r} A e^{-jkr} e^{j\omega r}$$

де A- деяка постійна; до - хвилеве число. Позначаючи $A = r_{ep} p_m$, де r_{ep} - середній радіус сфери; p_m амплітуда тиску на поверхні сфери, отримаємо

$$p = \frac{r_{ep} p_m}{r} e^{f(\omega t - kr)}$$

Дана рівність дає зв'язок між тисками на відстанях r_{ep} від початку координат. Коливання частинок середовища в сферичної хвилі відбувається по напрямку радіусів. Тому коливальну швидкість в сферичної хвилі називають радіальної коливальної швидкістю. Визначаючи радіальну коливальну швидкість частинок з рівняння руху і підставляючи в результат p з виразу, отримаємо співвідношення між тиском і радіальної коливальної швидкістю в сферичної хвилі

$$\xi_r = \frac{1 + jkr}{j\omega pr} p$$

де ω - кругова частота. Рівняння дозволяє визначити хвильовий опір сферичних хвиль Z_a:

$$Z_a = \frac{p}{\xi_r} = \frac{j\omega kr}{1 + jkr}$$

Таким чином, хвильовий опір сферичних хвиль виражається комплексною величиною.

В умовах усталеного руху лінія течії співпадає повинна співпадати з траєкторією частинки і форма залишається незмінною з часом. При неусталеному русі дотичні до лінії течії виступають швидкості різноманітних частинок, отже в даному випадку лінії

течії і траєкторії не можуть співпадати. Якщо в рухомій рідині взяти нескінченно малий поперечний переріз і через всі його точки провести лінії течії, то утвориться трубчаста поверхня, яка називається трубкою течії. Частина потоку, укладена всередині трубки течії називається елементарною струминкою. Сукупність елементарних струминок, що представляють собою неперервну масу частинок, які рухаються по якому-небудь напрямку, створюють потік рідини.

2. АНАЛІЗ ПРОБЛЕМИ РОЗРОБКИ ІНСТРУМЕНТАЛЬНОГО ЗАСОБУ

2.1 Коливальний процес системи з розподіленими сталими

На практиці досить часто зустрічаються коливальні системи з розподіленими постійними. Якщо маси, пружності і активні опори будь-яких рівних ділянок системи однакові, то її називають системою з рівномірно розподіленими постійними. До такої системи може бути віднесена однорідна електрична лінія, кожна ділянка довжини якою володіє індуктивністю, ємністю і активним опором, довгий однорідний стрижень.

Якщо ділянки коливальні системи мають переважно властивостями одного з трьох основних елементів, то таку систему називають системою з нерівномірно розподіленими постійними, Ці системи при коливанні мають безліч власних частот. Як приклад можна привести стрижень з нерухомо закріпленим кінцем при зосередженої сили, прикладеної до вільного кінця, або натягнуту круглу мембрану з рівномірно розподіленим по ній акустичним тиском.

При поперечних коливаннях закріпленого в опорі стрижня пружний вигин позначається в основному в місці кріплення стрижня в опорі. Для обчислення параметрів, наприклад частоти коливання, системи з нерівномірно розподіленими постійними можна замінити еквівалентними системами із зосередженими постійними.

Всі реальні коливальні системи мають велике число ступенів свободи. Вони складаються з окремих елементів, які можуть бути розглянуті як окремі матеріальні точки, здатні зміщуватися відносно один одного. Наприклад, що розгойдуються гойдалки, крім поступального руху, можуть здійснювати обертальний рух, а їх складові елементи будуть вібрувати. З усіх цих рухів можна виділити основний ряд коливань. Таким чином, велика кількість ступенів свободи реальної механічної системи можуть бути приведені до оптимальної кількості.

Коли кожен з складових механічної систему елементів володіє одним ступенем свободи, то її можна розглядати як деяку кількість пов'язаних між собою коливальних систем. Сенс такого розгляду полягає в тому, що, знаючи характер коливань кожної системи з одним ступенем свободи, можна встановити, як зміниться її поведінка внаслідок їх пов'язаності між собою.

Пов'язаною називається система, що складається з декількох простих систем, з'єднаних між собою таким чином, що рух однієї з них викликає силу, діючу на всі інші. При русі однієї з мас з'єднують їх пружини деформуються, і з'являється сила, яка діє на інші маси.

2.2 Опір плоских та сферичних хвиль

Акустична хвиля, як було сказано раніше, виникає в середовищі при коливаннях механічних систем. Тіло, що коливається в пружною середовищі, є джерелом акустичних хвиль. Порушуючи поздовжні коливання, наприклад, в сталевих струнах, можна викликати звучання цих струн. Тут вагається тілом, що випромінюють акустичні коливання, служить струна, поєднана з корпусом музичного інструменту. Телефони чи гучномовці створюють акустичні коливання, перетворюючи електричну енергію змінного струму в акустичну. Тут коливальним тілом виступає мембрана телефону або дифузор гучномовця, В гідроакустичних пристроях джерелом акустичної енергії служить гідроакустичний перетворювач, що випромінює поверхня якого пов'язана зі спеціальним електромеханічним перетворювачем (активним елементом), де відбувається перетворення електричної енергії в механічну. Коливання передаються частинкам навколишнього середовища, закон зміни щільності яких буде повторювати закон зміщення випромінюваної поверхні або закон зміни електричного поля в активному елементі гідроакустичного перетворювача. Так відбувається процес випромінювання акустичної енергії. В шарі середовища, що прилягає до поверхні, що випромінює, створюється змінне стиснення і розрідження—акустичний тиск. Акустичний тиск

буде впливати і на поверхню джерела акустичних хвиль. Це так звана реакція середовища на випромінювач.

Для того щоб створити процес випромінювання, випромінювач повинен мати певну потужністю, що йде на перетворення електричної енергії в механічну і подолання реакції середовища. Визначимо силу, яку необхідно прикласти до поверхні, що випромінює, щоб врівноважити реакцію середовища. Для цієї мети візьмемо поршневий випромінювач! і помістимо його в трубу з водою (рис. 43). Під поршневим ми розуміємо випромінювач, всі крапки поверхні, що випромінює якого мають в один і той же час однакові зміщення. При роботі поршня в трубі створюється плоска хвиля, що біжить (вважаємо, що відбиття від протилежного При випромінюванні на низьких частотах опір з боку середовища полягає в тому, що випромінююча поверхня при своєму русі як би розгойдує прилеглий до неї шар. Якщо ж довжина хвилі менше розмірів даного об’єкта, який має місце, в основному на високих частотах, то опір випромінювання середовища визначається цілком деформацією середовища - стисненням чи розрідженням, або активним опором випромінювання. Реактивний опір випромінюючи ня при цьому не виявляється.

3. ЗАСОБИ РОЗРОБКИ

3.1 Середовище розробки Visual Studio

Microsoft Visual Studio – один з продуктів компанії Microsoft, що включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших функцій та інструментальних засобів. MS Visual Studio .NET – це набір інструментів і засобів розробки різного роду застосувань та сервісів. MS Visual Studio - це мультипрограмне середовище, яке підтримує багато мов програмування, але найчастіше використовують C++ та C#. У Visual Studio кожне окреме застосування є рішенням, що складається з одного чи декількох проектів. Одночасно можна відкрити тільки одне рішення. Інтегроване середовище розробки включає в себе шаблони проектів найбільш поширених типів. Завдяки шаблонам і проектам користувач може зосередитись над реалізацією окремої функції в той час як проект буде виконувати загальне управління та завдання побудови (рисунок 3.1)

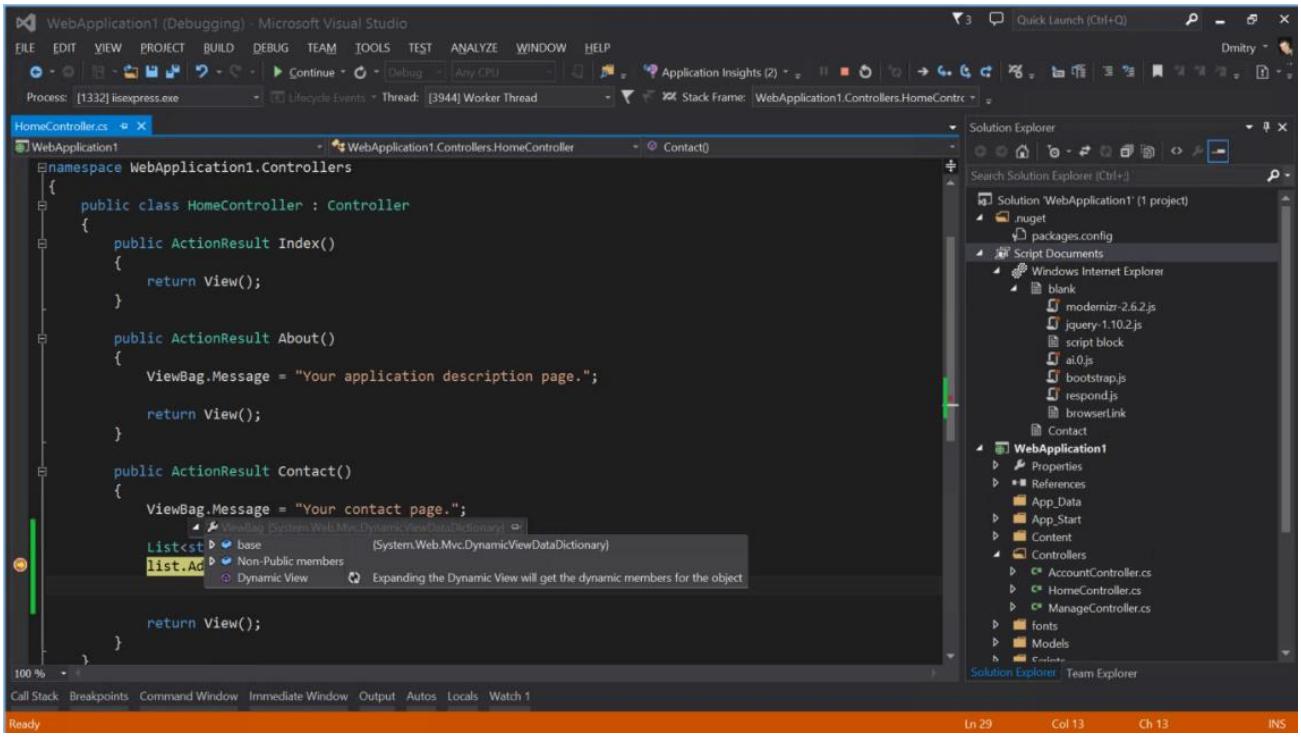

Рисунок 3.1 – стандартний робочий простір Visual Studio

Для створення нового проекту використовується майстер застосувань. Майстер створення програм надає користувальницький інтерфейс для створення проекту за шаблоном та створення шаблонів для файлів вихідних текстів. Майстер налаштовує структуру програми, основні меню і панелі інструментів, забезпечує включення деяких заголовних файлів.

3.2 Мова програмування C#

C# — об’єктно-оріентовна мова програмування, яка відноситься до мов з с-подібним синтаксисом. C# розроблялась як мова програмування прикладного рівня для Common Language Runtime(CLR) і, як такий, залежить, перш за все, від можливостей самої CLR. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи можуть конкретні особливості мови бути трансльовані у відповідні конструкції CLR. Так, з розвитком CLR значно збагатився і сам C#.

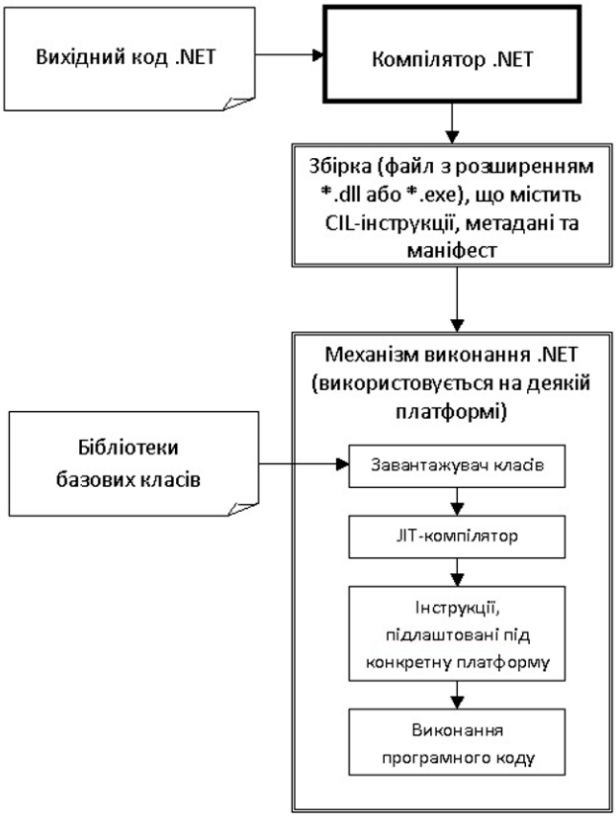

Рисунок 3.2- Схема взаємодії між вихідним кодом, компілятором .NET та механізмом виконання .NET

3.3 Об'єктно-рольове моделювання

У словнику даних розробник може зберігати відомості про рішеннях, прийнятих в процесі проектування бази даних. Таким чином, вивчення семантичного моделювання може виявитися надзвичайно корисним для проектування системи управління словником даних, оскільки в ньому можуть бути вказані типи сутностей, які словник повинен підтримувати і описувати, наприклад, категорії сутностей (такі як звичайні і слабкі сутності в ER-моделі), правила цілісності (такі як повне або часткове участь в зв'язку ER-моделі), супертипи і підтипи сутностей і т.д.

ORM моделювання називається також моделюванням на основі фактів, оскільки проектувальник починає процес моделювання з записи (природною мовою або за допомогою спеціальних графічних позначень) ряду елементарних фактів (або, точніше, типів фактів), які в сукупності характеризують всі особливості модельованої організації. Нижче наводяться деякі приклади таких типів фактів.

—Кожен працівник має не більше одного імені.

—Кожен працівник звітує про свою діяльність не більш ніж перед одним працівником.

—Якщо працівник el звітує перед працівником e2, то зворотне (тобто працівник e2 звітує перед працівником el) неможливо.

—Жоден працівник не може одночасно керувати проектом і оцінювати результати його виконання.

Очевидно, що типи фактів насправді представляють собою зовнішні предикати або бізнес-правила. Як випливає з назви цієї статті, підхід до проектування бази даних в ORM-моделюванні дуже близький до тих методів, які вважають за краще прихильники використання бізнес-правил і сам автор. У загальному випадку факти являють собою ролі, виконувані об'єктами в зв'язках (звідси і назва "об'єктно-рольове моделювання"). Зверніть увагу на те, що об'єкти в даному контексті є сутності, а не об'єкти в конкретному сенсі цього терміну, а зв'язки між ними не обов'язково є

двосторонніми. Однак факти дійсно елементарні, тобто їх декомпозиція на менші факти неможлива.

Заслуговує уваги те, що в ORM-моделюванні немає поняття атрибут. В результаті, як показано в цій статті, ORM-проекти концептуально простіше і стійкіше, ніж їх аналоги, виконані за методом RM-моделювання. Але атрибути можуть і повинні з'являтися в проектах на основі ER-моделі або визначеннях SQL, які виробляються (автоматично) на підставі проекту ORM.

У ORM-моделюванні також особливо підкреслюється використання фактів-зразків (тобто примірників фактів-зразків, які інакше можна було б назвати висловлюваннями) в якості способу перевірки коректності проекту з боку кінцевого користувача. Як стверджується в даній статті, такий підхід досить просто реалізувати за допомогою моделювання на основі фактів, але набагато важче здійснити при роботі з ER-моделлю.

Безумовно, існує багато логічно-еквівалентних способів опису одного і того ж підприємства і, відповідно, багато логічно-еквівалентних ORM-схем. З цієї причини ORM-моделювання включає набір правил перетворення, що дозволяють виконувати перетворення з отриманням логічно-еквівалентних схем.

Тому ORM-інструменти дозволяють виконувати деяку оптимізацію проектів відповідно до вимог розробника. Як вже говорилося вище, з їх допомогою можна також виробляти схему у вигляді ER-моделі або визначення SQL на підставі схеми ORM, а також виконувати зворотну процедуру відновлення схеми ORM з існуючої схеми ER або SQL.

Залежно від використовуваної СУБД, створена схема SQL може включати декларативні обмеження в стилі SQL; інший варіант може передбачати реалізацію таких обмежень за допомогою тригерів або збережених процедур. Щодо обмежень відзначимо, що ORM-модель, на відміну від ER-моделі, за визначенням містить розвинену мову для визначення обмежень.

3.4 Платформа NET Framework

NET Framework - програмна платформа, розроблена компанією Microsoft в 2002 році. Основу платформи складає загальномовне середовище виконання Common Language Runtime (CLR), яке використовується для різних мов програмування. Усім мовам програмування, що використовують цю середу, доступні функціональні можливості CLR.

Хоча .NET запатентована технологія корпорації Microsoft, а тому офіційно розрахована лише на використання під ОС сімейства Microsoft Windows, але існують незалежні проекти, такі як Mono та Portable.NET. Вони в свою чергу дозволяють запускати програми .NET на інших операційних системах. В наш час платформа .NET Framework поширюється у вигляді .NET Core, яка передбачає собою кросплатформену розробку та експлуатацію.

По-перше, в основі розробці .NET Framework покладено забезпечення свободи розробника надавши йому можливості створення додатків різноманітних типів. Вони в свою чергу можуть виконуватися на різних пристроях та в будь-яких середовищах. По-друге, в основу вкладена орієнтація на системи під ОС Microsoft Windows.

У програмі для .NET Framework, що написана на будь-якій підтримуваній мові програмування, весь код транслюється компілятором в єдиний проміжний байт-код для .NET Common Intermediate Language (CIL), а потім виходить збірка assembly. Після чого код або виконується віртуальною машиною, або за допомогою NGen.exe транслюється утилітою в виконуваний код для заданого цільового процесора. Переважає використання віртуальної машини, оскільки вона може позбавити розробника від необхідності знати про особливості апаратної частини. При використанні CLR вбудований в неї JIT-компілятор транслює проміжний байт-код в машинний код для конкретного процесора. Сучасні технології дозволяють досягти високого рівня швидкодії завдяки динамічній компіляції. Віртуальна машина CLR також автоматично конфігурує базову безпеку, управління пам'яттю у системі винятків, що в свою чергу позбавляє розробника від частини роботи.

Entity Framework представляє собою спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. На відміну від ADO.NET, яка дозволяє створювати підключення, команди та інші об'єкти для взаємодії з базами даних, Entity Framework має більш високий рівень абстракції, яка в свою чергу дозволяє абстрагуватися від самої бази даних і працювати з даними в ній незалежно від сховища та його типу. На фізичному рівні розробник оперує таблицями, індексами, первинними і зовнішніми ключами, а на концептуальному рівні запропонований Entity Framework, він вже має справу з об'єктами. У першій версії Entity Framework - 1.0 2008 року була дуже обмежена функціональність, базова підтримка ORM і лише один підхід до зв’язку з БД - Database First. Після релізу версії 4.0 2010 року Entity Framework перетерпіла багато змін - з цього часу Entity Framework стала провідною технологією для доступу до даних. З нововведень у фреймворк були вписані нові можливості взаємодії з БД - методи Model First і Code First. І нарешті, Entity Framework 6.0 що побачив світ у 2013 році отримав можливість асинхронного доступу до даних.

У центральній концепції Entity Framework полягає поняття сутності або entity. Вона представляє собою набір даних, асоційованих з певним об'єктом. Тому представлена технологія передбачає роботу не з таблицями, як було раніше, а з об'єктами і їх наборами.

Будь-який entity, як і об'єкт у реальному світі, передбачає собою низку властивостей. Наприклад, опишемо сутність людини, в ній ми можемо відокремити ось такі властивості: ім'я, прізвище, зріст, вік, вага. Вони необов'язково представляють прості дані типу integer, а й можуть переходити у більш комплексні структури даних. У представлених сутностей може бути одна або декілька властивостей, які відрізняють конкретну сутність від інших і будуть унікально представляти цю сутність. Усі ці властивості іменують ключами.

При цьому entity можуть бути пов'язані асоціативним зв'язком один-до-многих, один-до-одного і багато-до-багатьох, у подібності до того, як в різних базах даних представлений зв'язок через зовнішні ключі.

Також треба відмітити рису Entity Framework як запити LINQ що використовуються для вибірки даних з БД. LINQ дозволяє не тільки отримувати певні рядки, що зберігають об'єкти з BD, а й отримувати дані, що пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Дана модель зіставляє класи сутностей з представленими таблицями в БД.

Entity Data Model поділяється на три рівні : концептуальний, рівень сховища і рівень зіставлення або мапінга.

Концептуальний рівень — відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища — визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних.

Рівень зіставлення — служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, розробник має можливість через класи, визначені у програмному продукті, взаємодіяти з таблицями у базі даних.

3.5 Технологія для розробки інтерфейсу

Для створення клієнтської частини інтерфейсу були використані такі технології як Windows Forms. Він представляє собою інтерфейс для програмування додатків, що відповідає за графічний інтерфейс користувача а також входить до збірки Microsoft .NET Framework (Рисунок 3.3).

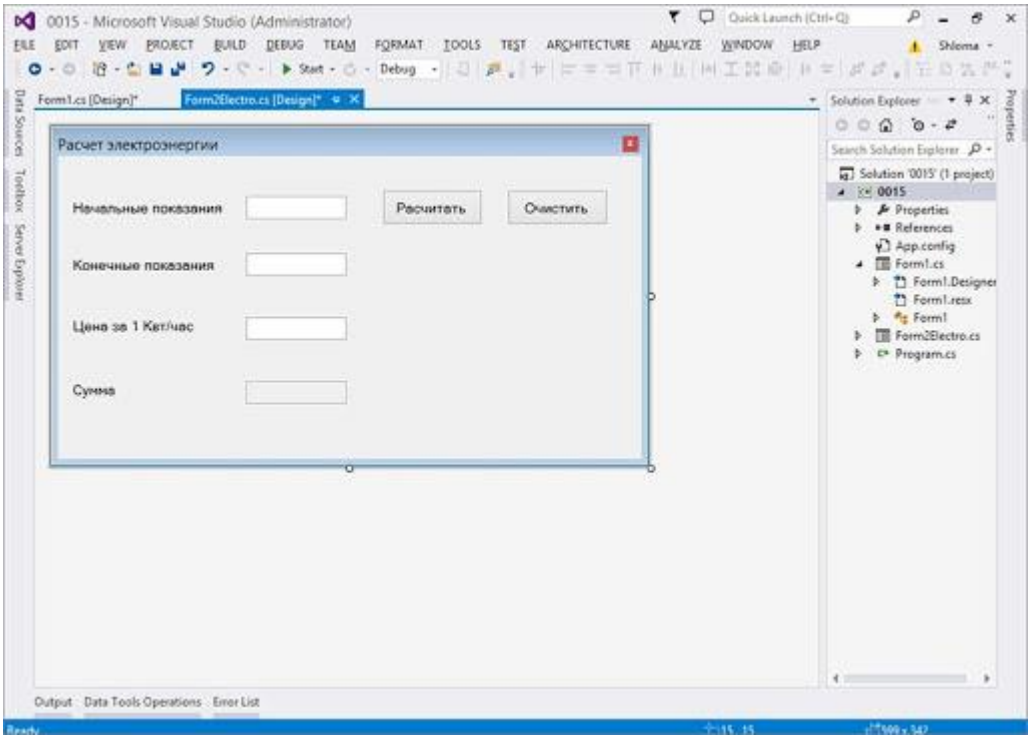

Рисунок 3.3 – стандартний робочий простір Windows Forms

Завдяки інтерфейсу користувач має спрощений доступ до елементів інтерфейсу Microsoft Windows за рахунок створеної обгортки для існуючого Win32 API в керованому коді. Причому керований код – це сукупність класів, які реалізують API для Windows Forms, що в свою чергу не залежать від мови розробки. Windows Forms надає можливість розробки суміжного графічного інтерфейсу для користувача. Однак цей інтерфейс є лише обгорткою Windows API-компонентів, і ряд її методів здійснюють прямий доступ до Win32-функцій зворотного виклику, тому і доступні лише на Windows платформі.

3.6 Середовище Microsoft SQL Server

Microsoft SQL Server система управління базами даних, розробки Microsoft. Дана система використовує версію SQL як мову запитів, що отримала назву Transact-SQL, реалізована як SQL-92 з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL

Server та Sybase ASE для взаємодії з мережею використовують протокол передачі табличних даних. Microsoft SQL Server також підтримує Open Database Connectivity - інтерфейс взаємодії застосунків з СУБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, з'єднуватися з SQL Server.

SQL Server здійснює підтримку надлишкового дублювання даних за декількома сценаріями. Сервер робить «знімок» бази даних та відправляє його одержувачам. Усі зміни бази даних відправляються користувачам водночас. Синхронізація бази даних декількох серверів. Усі зміни баз даних незалежні на кожному сервері, але звірка даних відбувається під час синхронізації. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних. В SQL Server 2005 вбудована підтримка .NET Framework. Завдяки цьому, збережені процедури бази даних, можуть бути написані на будь-якій мові платформи .NET а також використовувати повний набір бібліотек, доступних .NET Framework(рисунок 3.4).

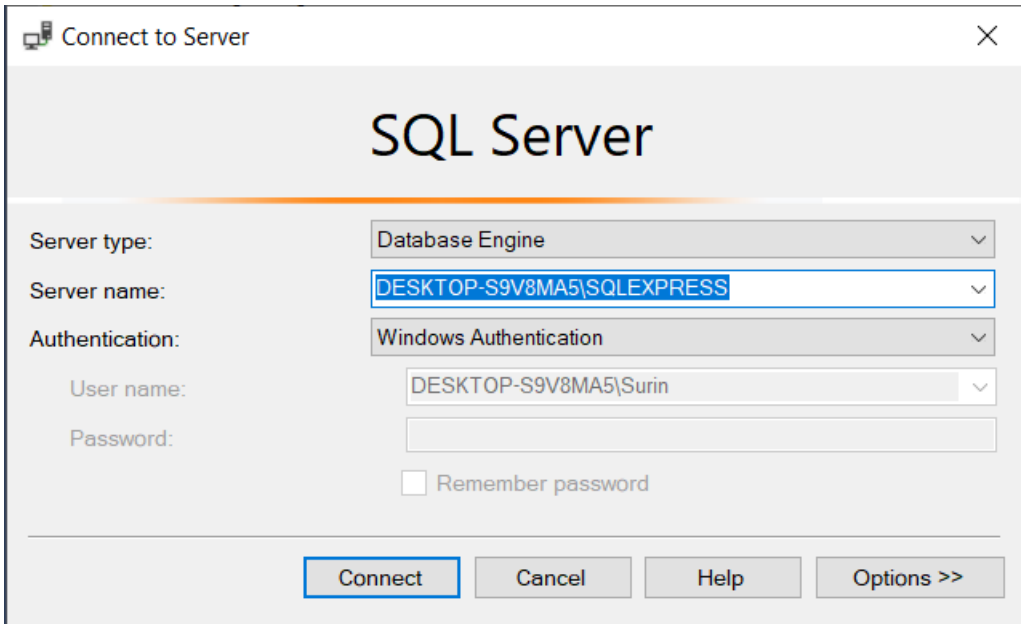

Рисунок 3.4-підключення до SQL Server

На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, без використання вбудованих засобів Windows.

Це в свою чергу значно підвищує продуктивність в порівнянні зі звичайними алгоритмами Windows.

У SQL Server упор робиться на спрощення конфігурацією та управління СУБД, а також полегшення масштабованості. Зокрема, розробники заявляють, що процес, який включає в себе мільйони колонок даних, завдяки покращенню масштабованості зможе виконуватися впродовж декількох секунд, а не хвилин. Також, у новій версії SQL Server планувалося включення функцій для спрощення створення сховищ даних і їх управління, а також виконання операцій, пов'язаних з інтелектуальною підтримкою бізнесу. Розробники Microsoft обіцяють також новий API, з підтримкою на платформі .NET. тим самим дане впровадження позбавляє від необхідності використовувати специфічний код DTS.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Архітектура бази даних

При організації роботи з БД в першу чергу треба забезпечити незалежність програм від даних. Це обумовлено тим, що при зміні системи, а також з метою забезпечення ефективного обслуговування користувачів необхідно виконувати роботи щодо зміни методів зберігання даних в БД, шляхів доступу до даних, змінювати структури і формати даних та зв'язки між ними. Якщо уникнути застосовування спеціальних підходів та при написанні додатку введення програмного опису засобів зберігання даних або форматів даних, то при внесені змін у БД для наступних випадків є необхідність корегувати сам текст програми користувача, а це в свою чергу потребує значних витрат у часі та ресурсах. Опис структури даних на будь-якому рівні називається схемою.

Існує три різних типи схем БД, які визначаються згідно з рівнями абстракції архітектури СУБД. На самому верхньому рівні є декілька зовнішніх схем, які відповідають різним представленням даних. Цей рівень визначає точку зору на БД окремих застосувань. Кожне застосування бачить і обробляє тільки ті дані, які необхідні цьому застосуванню. На рисунку 4.1 показана трирівнева модель архітектури СУБД (Рисунок 4.1).

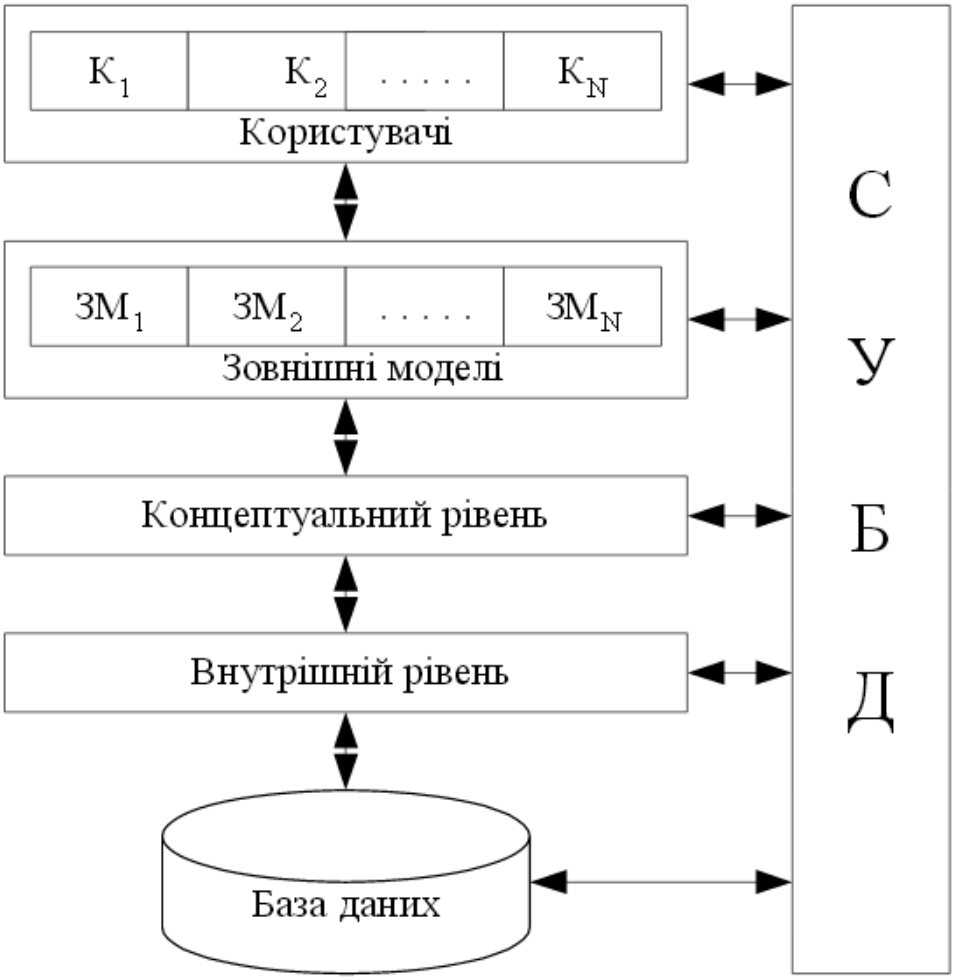

Рисунок 4.1 - трирівнева модель архітектури СУБД

Незалежність застосувань від збереженої інформації забезпечується засобами СУБД. В такому випадку користувачі при роботі з БД, не повинні знати внутрішнє представлення даних.

На концептуальному рівні опис БД називається концептуальною схемою. Тут БД представлена в найбільш загальному вигляді, яка об'єднує дані, що використовуються всіма застосуваннями, які працюють з БД. Фактично концептуальний рівень відображає модель предметної області, для якої створювалася БД.

На внутрішньому рівні опис БД називається внутрішньою схемою. Тут БД представлена у вигляді безпосередньо даних, що розташовані в файлах, які відповідають фізичній організації БД.

4.2 Технології розробки програмного продукту

4.2.1 Впровадження залежностей Dependency injection

Впровадження залежностей представляє механізм, який дозволяє зробити взаємодіючі в додатку об'єкти слабо зв'язаної. Такі об'єкти пов'язані між собою через абстракції, наприклад, через інтерфейси, що робить всю систему більш гнучкою, більш адаптованою і розширюється.

Нерідко для установки залежностей в подібних системах використовуються спеціальні контейнери - IoC-контейнери (Inversion of Control). Такі контейнери служать свого роду фабриками, які встановлюють залежності між абстракціями і конкретними об'єктами і, як правило, управляють створенням цих об’єктів. І якщо раніше в ASP.NET 4 і інших попередніх версіях треба було використовувати різні зовнішні IoC-контейнери для установки залежностей, такі як Ninject, Autofac, Unity, Windsor Castle, StructureMap, то ASP.NET Core вже має вбудований контейнер впровадження залежностей, який представлений інтерфейсом IServiceProvider. А самі залежно ще називаються сервісами, власне тому контейнер можна назвати провайдером сервісів. Цей контейнер відповідає за зіставлення залежностей з конкретними типами і за впровадження залежностей в різні об'єкти.

4.2.2 Стиль CQRS

CQRS - це стиль архітектури, в якому операції читання відокремлені від операцій записи. Підхід сформулював Грег Янг на основі принципу CQS, запропонованого Бертраном Мейером. Найчастіше (але не завжди) CQRS реалізується в обмежених контекстах (bounded context) додатків, що проектуються на основі DDD. Одна з природних причин розвитку CQRS - не симетричний розподіл навантаження і складності бізнес-логіки на read і write – підсистеми. Більшість бізнес-правил і складних перевірок знаходиться під write - підсистемі. При цьому читають дані часто в рази частіше, ніж змінюють(Рисунок 4.2).

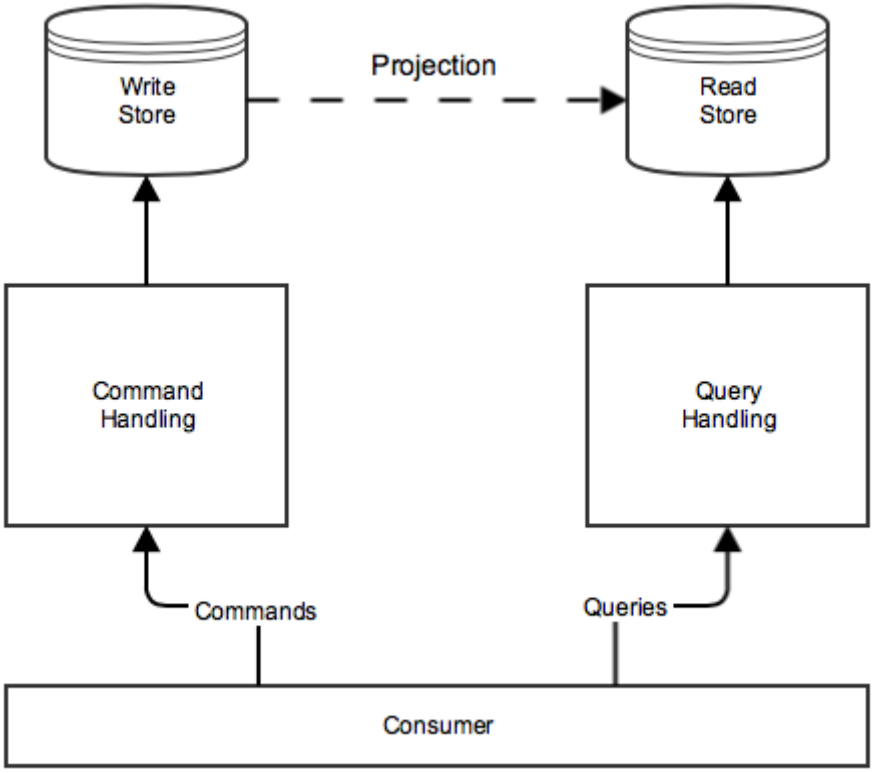

Рисунок 4.2 – вплив операцій read write на різні БД

4.2.3 Архітектура MVC

Одним з характерних моментів платформи ASP.NET Core є застосування патерну MVC. Причому остання версія MVC-фреймворка, який застосовується в ASP.NET Core, має номер 3.0 / 3.1. Тому важливо не плутати ASP.NET MVC 5, який застосовується в ASP.NET 4.5-4.8, і фреймворк MVC, який застосовується в ASP.NET Core. Хоча в багатьох аспектах ці фреймворки будуть збігатися.

Також невірно ототожнювати ASP.NET Core цілком з фреймворком ASP.NET Core MVC. Фреймворк ASP.NET Core MVC працює поверх платформи ASP.NET Core, і призначений для того, щоб спростити створення програми. Але ми можемо і не використовувати MVC, а застосовувати чистий ASP.NET Core і на ньому цілком вибудовувати логіку програми.

Сам патерн MVC не є якоюсь новою ідеєю в архітектурі додатків, він з'явився ще в кінці 1970-х років в компанії Xerox як спосіб організації компонентів в графічному додаток на мові Smalltalk.

Концепція патерна MVC передбачає поділ додатка на три компоненти. (Рисунок 4.3)

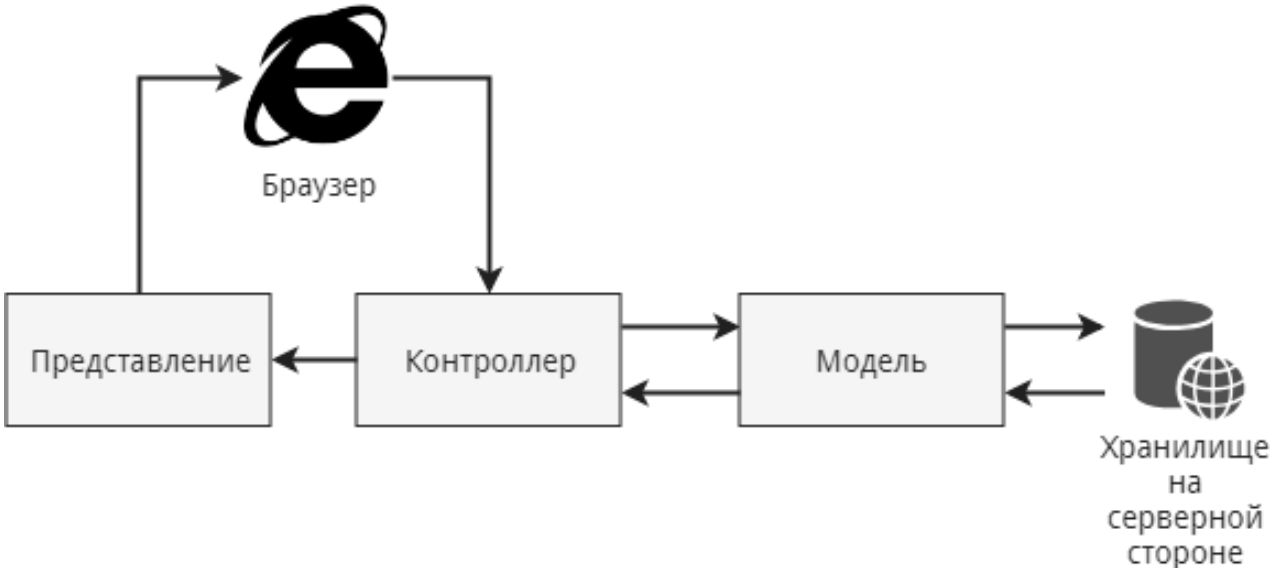

Рисунок 4.3 – Модель MVC

Модель (model): описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як правило, об'єкти моделей зберігаються в базі даних. У MVC моделі представлені двома основними типами: моделі уявлень, які використовуються уявленнями для відображення і передачі даних, і моделі домену, які описують логіку управління даними. Модель може містити дані, зберігати логіку управління цими даними. У той же час модель не повинна містити логіку взаємодії з користувачем і не має визначати механізм обробки запиту. Крім того, модель не повинна містити логіку відображення даних в поданні.

Подання (view): відповідають за візуальну частину або призначений для користувача інтерфейс, нерідко html-сторінка, через який користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час уявлення не повинно містити логіку обробки запиту користувача або управління даними.

Контролер (controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, поданням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує вводяться користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання, наповненого даними моделей.

У цій схемі модель є незалежним компонентом - будь-які зміни контролера або подання ніяк не впливають на модель. Контролер і уявлення є відносно незалежними компонентами. Так, з уявлення можна звертатися до певного контролеру, а з контролера генерувати уявлення, але при цьому нерідко їх можна змінювати незалежно один від одного.

4.2.3 Unit тести

Для створення юніт-тестів вибираються невеликі ділянки коду, які треба протестувати. Тестований ділянку, як правило, повинен бути менше класу. У більшості випадків тестується окремий метод класу або навіть частина функціоналу методу. Упор на невеликі ділянки дозволяє досить швидко писати простенькі тести.

Одного разу написаний код нерідко читають багато разів, тому важливо писати зрозумілий код. Особливо це важливо в юніт-тестах, де в разі невдачі при тестуванні розробник повинен швидко прочитати вихідний код і зрозуміти в чому проблема і як її виправити. А використання невеликих ділянок коду значно спрощує подібну роботу. При тестуванні важливо ізолювати тестований код від решти програми, з якою він взаємодіє, щоб потім чітко визначити можливість помилок саме в цьому ізольованому коді. Що спрощує і підвищує контроль над окремими компонентами програми.

Для написання юніт-тестів ми можемо самі створювати весь необхідний функціонал, використовувати якісь свої способи тестування, однак, як правило, для цього застосовуються спеціальні фреймворки. Деякі з них:

—xUnit.net: фреймворк тестування для платформи .NET. Найбільш популярний фреймворк для роботи саме з .NET Core і ASP.NET Core
—MS Test: фреймворк юніт-тестування від компанії Microsoft, який за замовчуванням включений в Visual Studio і який також можна використовувати з .NET Core
—NUnit: портований фреймворк з JUnit для платформи .NET
Розробка через тестування представляє процес застосування юніт-тестів, при якому спочатку пишуться тести, а потім вже програмний код, достатній для виконання цих тестів.

Використання TDD дозволяє знизити кількість потенційних багів в додатку. Створюючи тести перед написанням коду, ми тим самим описуємо спосіб поведінки майбутніх компонентів, не зв'язуючи себе при цьому з конкретною реалізацією цих тестованих компонентів (тим більше що реалізація на момент створення тесту ще не існує). Таким чином, тести допомагають оформити і описати API майбутніх компонентів.

4.3 Концептуальна модель

Концептуальна модель — модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.
Неформалізований опис розроблюваної імітаційної моделі включає визначення основних елементів системи, що моделюється, їх характеристики і взаємодія між елементами власною мовою. При цьому можуть використовуватися таблиці, графіки, діаграми і т.д. Неформалізований опис моделі необхідний як самим розробникам під час перевірки адекватності моделі, її модифікації, так і для взаєморозуміння з працівниками в інших галузях.

Концептуальна модель повинна містити вихідну інформацію для системного аналітика, що в свою чергу виконує формалізацію системи і використовує певні

методології та технології, тобто на основі неформалізованих описів здійснюється проведення більш суворого і докладного формалізованого опису.

Потім представлений формалізований опис транслюється в програму-імітатор відповідно до конкретного методу або технології програмування. Концептуальна модель створеного програмного продукту представлена у вигляді 5 таблиць із вказаними даними. Вона відображена на рисунку 4.2.

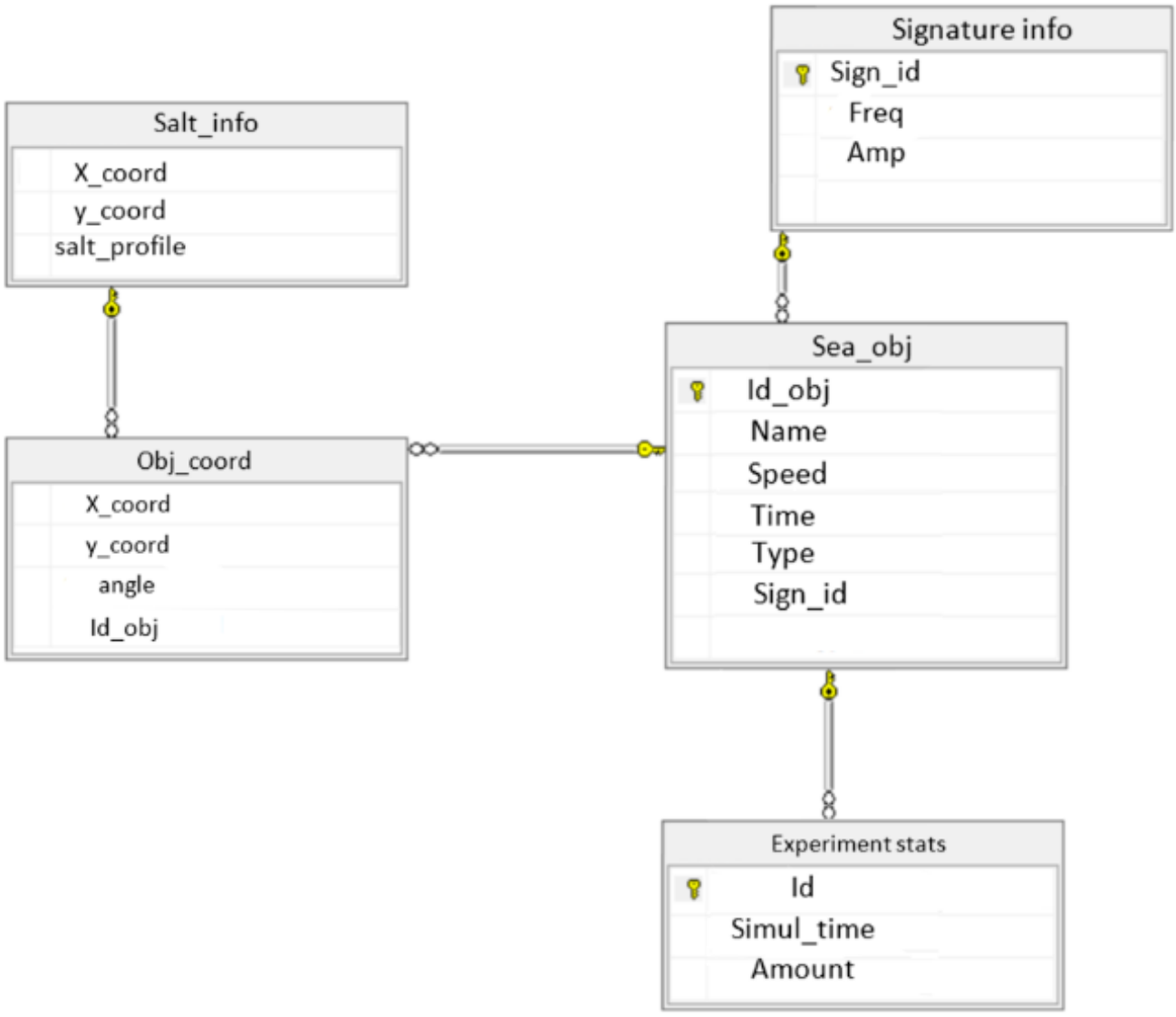

Рисунок 4.2 — Концептуальна модель

4.4 Інтерфейс користувача

Інтерфейс програмного продукту включає вікно авторизації та форму з вкладками, які забезпечують можливість додавання до бази даних початкових умов гідроакустичного експерименту.

Рівень представлення - рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти для користувача інтерфейсу, механізм отримання введення від користувача. Досягнуто збільшення гнучкості завдяки спрощенню і реалізації тільки необхідного інструментарію. Даний підхід полегшить розуміння програмного продукту зі сторони користувача та надає змогу працювати лише з тим функціоналом, який потребує користувач.

4.5 Структура таблиць бази даних

База даних представляє собою структуру з 5 зв’язаних між собою таблиць у середовищі MS SQL(Рисунок 4.4)

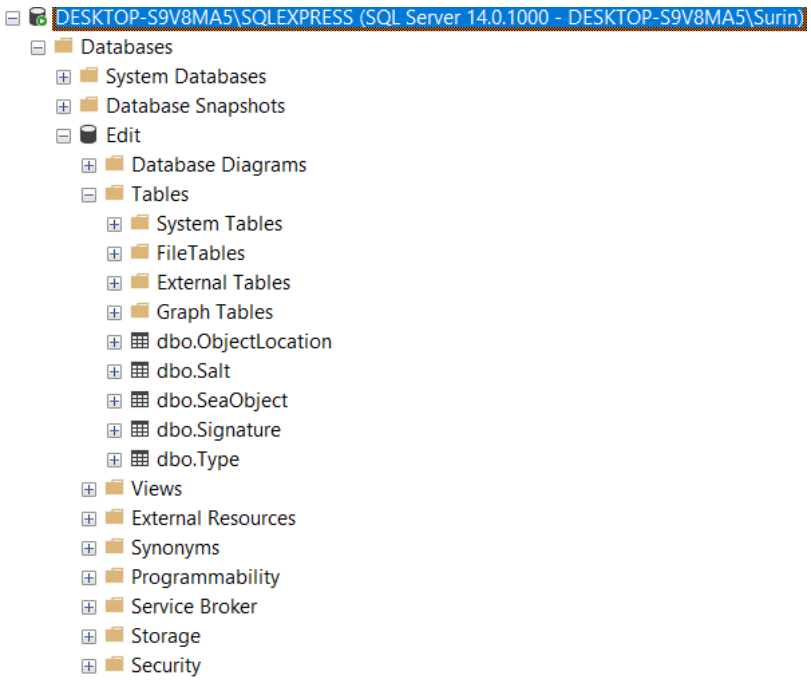

Рисунок 4.4 — База даних в середовищі MS SQL

У таблиці Salt_info (рисунок 4.5) представлені поля в які користувач вводить координати та значення солоності в певних секторах

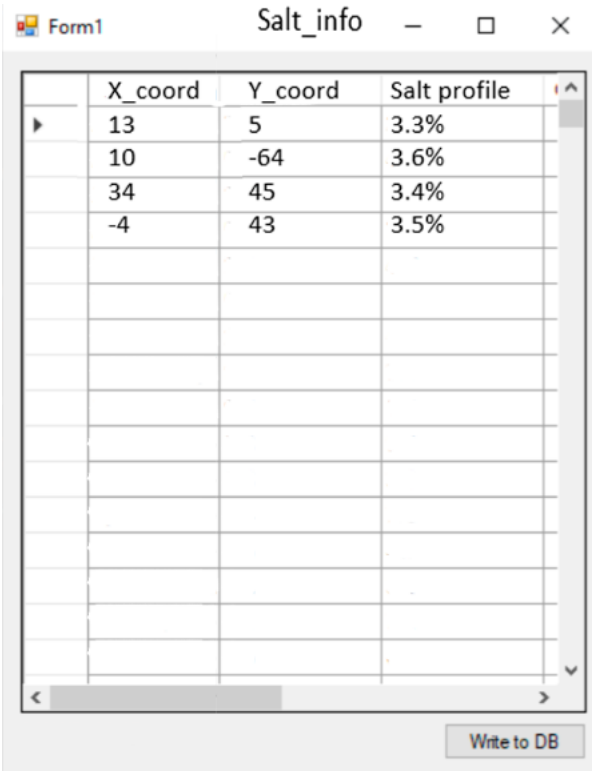

рисунок 4.5-Таблиця солоності

Таблиці Sea_obj е головною (рисунок 4.6), так як представляє основні об’єкти та їх характеристики на мапі.

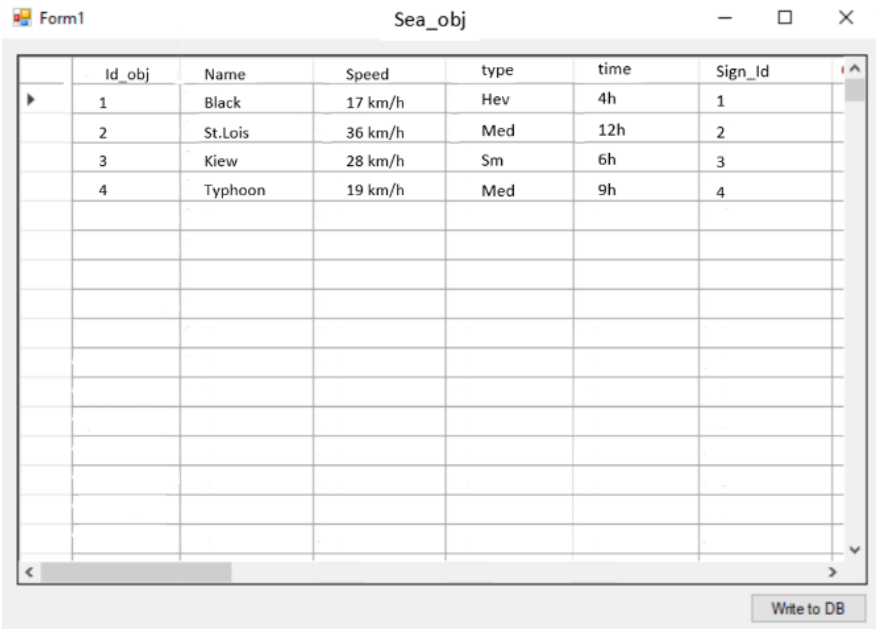

рисунок 4.6-Таблиця морських об’єктів

Поля таблиці мають такі значення Id_obj- ідентифікатор об’єкта, Name- його ім’я, Speed – швидкість руху, Time- час руху, Type-тип об’єкта та Sign_id -ідентифікатор сигнатури.

Таблиця Experiment stats (рисунок 4.7), зберігає дані про час симуляції експерименту та кількість ітерацій у сценарію

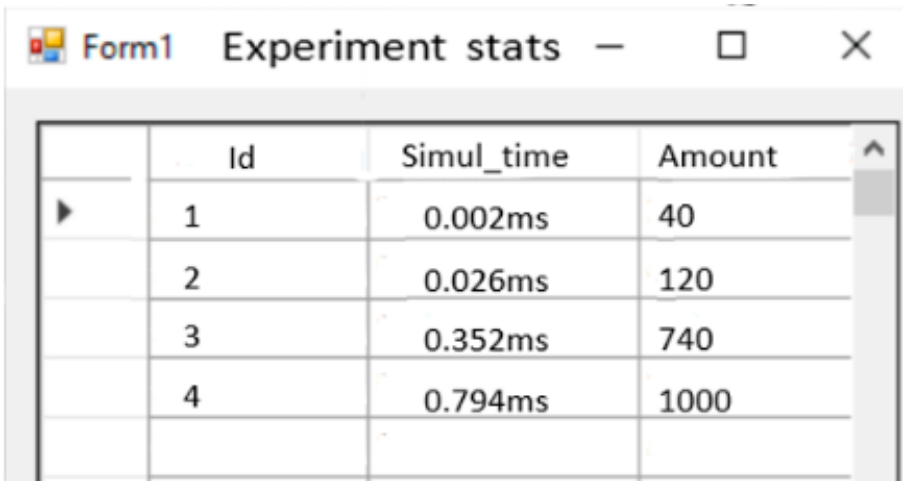

рисунок 4.7- Таблиця Experiment stats

Таблиця Signature info (рисунок 4.8), зберігає такі дані Sign_id- ідентифікатор сигнатури Freq-частота коливань, Amp-амплітуда коливань

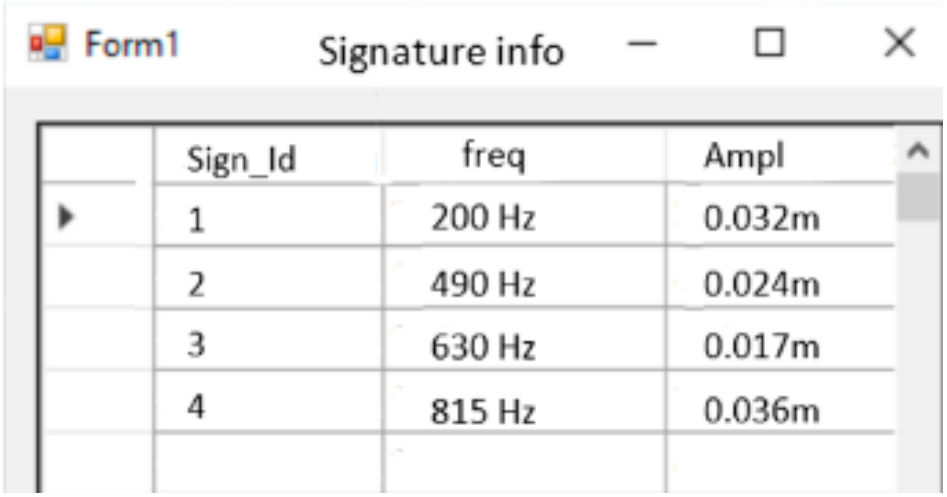

рисунок 4.8-Таблиця Signature info

У таблиці Obj_coord (рисунок 4.9) представлені поля в які користувач вводить координати та значення солоності в певних секторах X_coord та Y_coord – координати відносно центра, angle- кут, відносно якого рухається сам об’єкт, Id_obj- ідентифікатор об’єкта

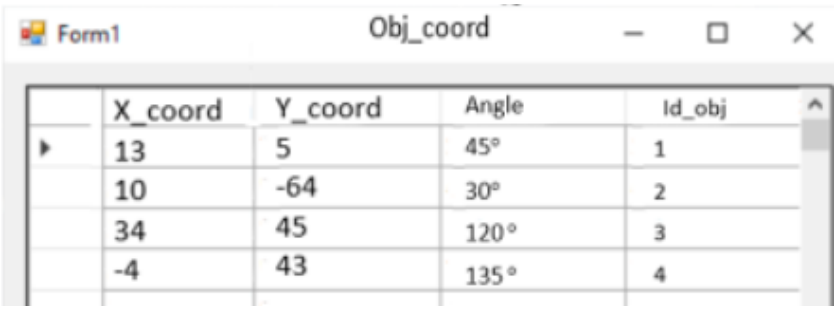

рисунок 4.9-Таблиця Obj_coord

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Системні вимоги та інсталяція

Для коректного запуску програмного продукту необхідна наявність встановлених пакетів платформи .NET Framework, версії не нижче 4.0.

Програмний продукт потребує встановлення на локальну машину, може бути запущений з використанням будь-якого носія при наявності певних додатків.
Для коректної роботи додатку апаратного забезпечення мусить задовольняти наступним вимогам:
— процесор Intel Core i3 2.4 GHz;
— оперативна пам’ять в об’ємі 4 Гб:
— відео пам'ять R7 360 2 Гб.

Для роботи з базою даних треба мати встановлений на апаратну частину MS SQL
Даний додаток має ряд обмежень:
—Максимальна кількість ядер процесора: 4;
—Максимальний розмір бази даних: 10 ГБ;
—Максимальний розмір пулу буфера на екземпляр бази даних: 1410 МБ.

5.2 Сценарій роботи з програмним продуктом

Для початку роботи з додатком необхідно запустити додаток, який у свою чергу зберігає усі модулі програмного комплексу. При запуску користувачеві відображається наступне вікно(рисунок 5.1).

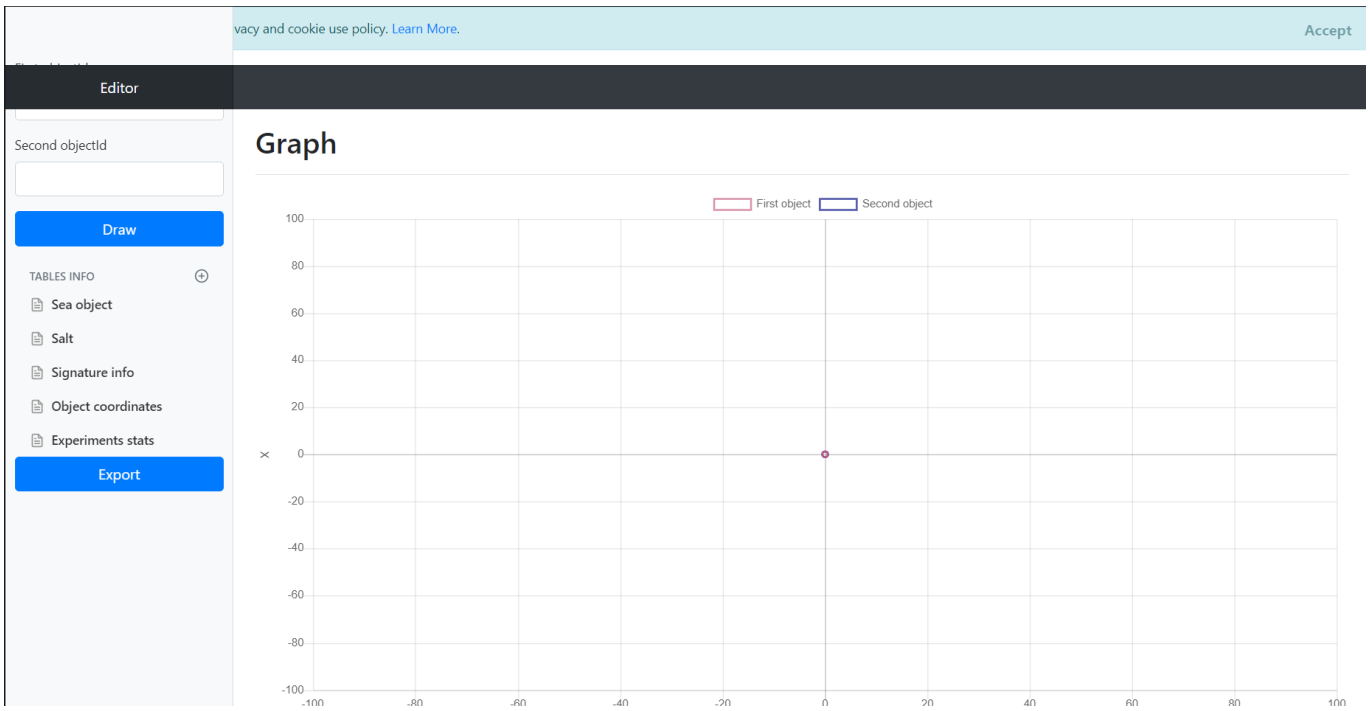

рисунок 5.1 – стартове вікно

Для початку треба ввести коректні дані у наявні поля для морських об’єктів, а саме назву, тип, швидкість, початкові та кінцеві координати і кут відносно центру координат. Після того, як користувач ввів потрібну інформацію та підтвердив свою дію – графічний інтерфейс виводить морські об’єкти на екран(Рисунок 5.2), а самі дані зберігаються у відповідних таблицях.

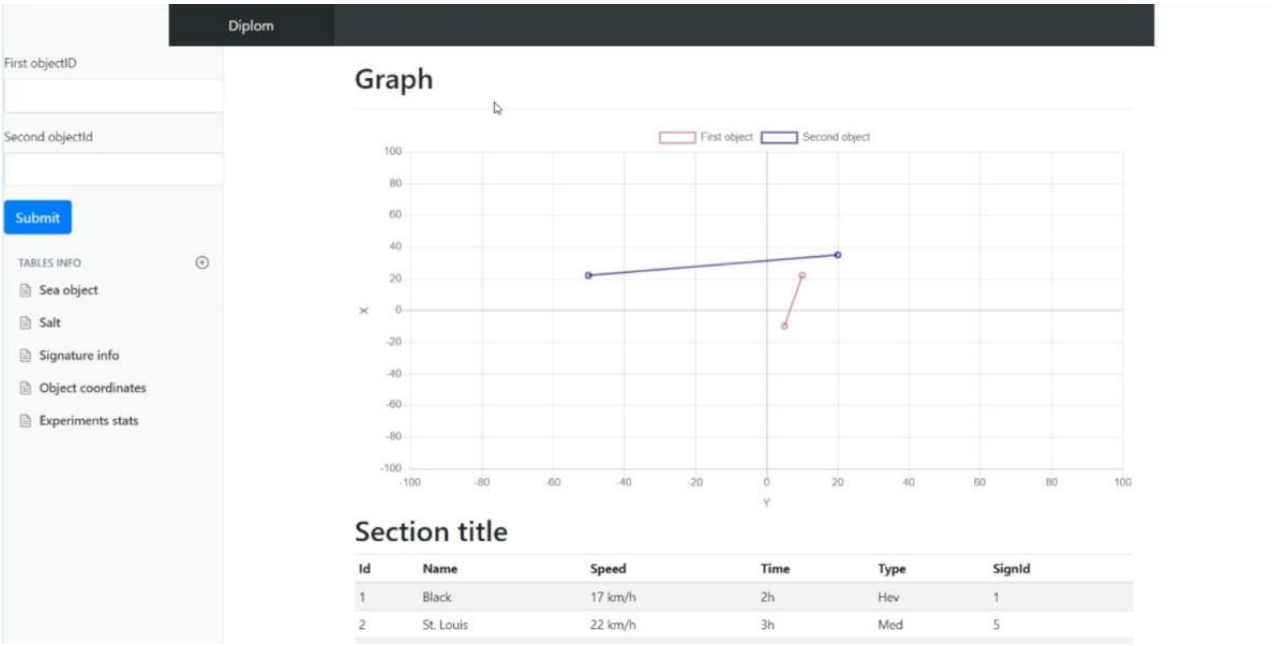

Рисунок 5.2-вивід на екран двох об’єктів

При роботі з графічним інтерфейсом виникає необхідність зміни параметрів координат точок на графічній панелі. Тому в веб-редакторі знизу таблиці розташована панель редагування даних з активної таблиці(Рисунок 5.3)

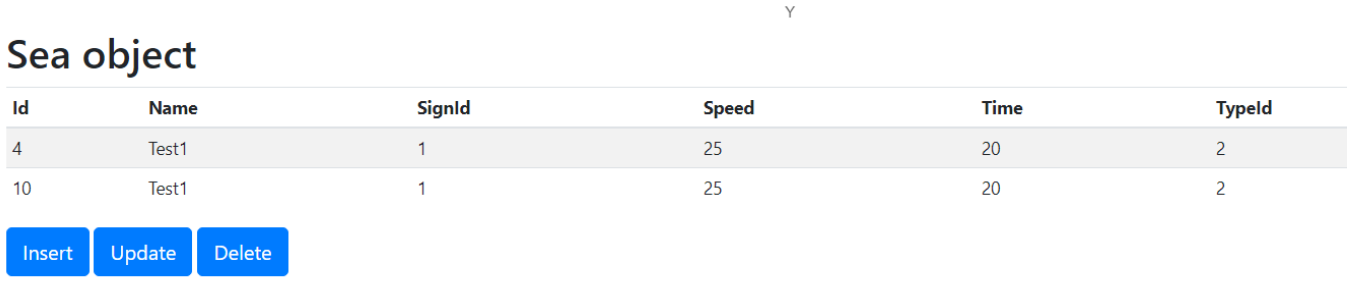

Рисунок 5.3-поле роботи з даними в таблиці

Усі характеристики об’єктів зберігаються у відповідній таблиці Object Coordinates(Рисунок 5.4).

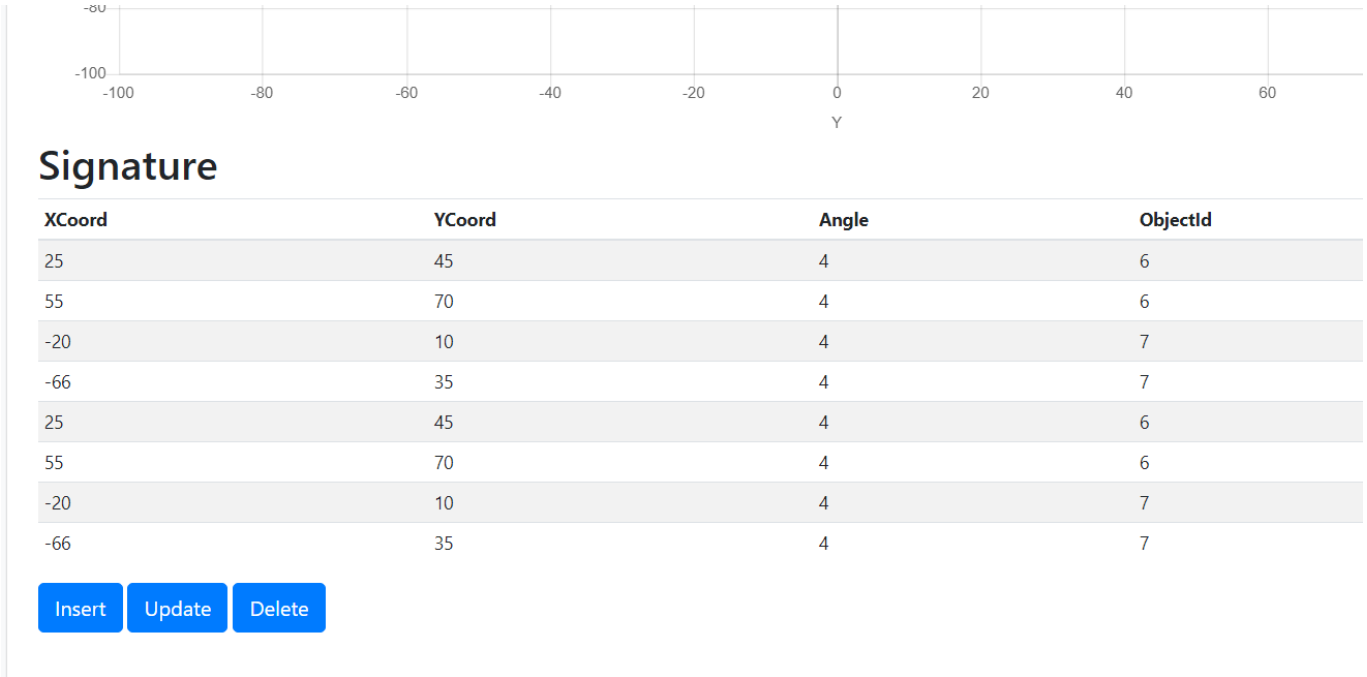

Рисунок 5.4-збереження характеристик об’єктів

Для того щоб користувач почав працювати з іншою таблицею, він повинен вибрати відповідну таблицю у вікні з лівого боку додатку(Рисунок 5.5).

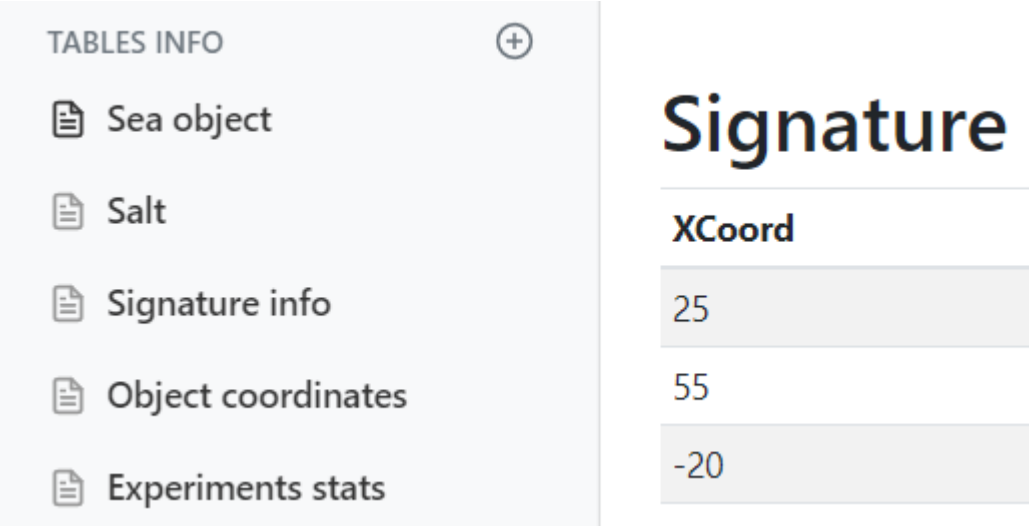

Рисунок 5.5-вибір активної таблиці

Для подальшої роботи зі сценарієм користувачу потрібно завантажити дані з таблиць у форматі Json. Вони в свою чергу будуть використовуватись в наступному модулі для розрахунків. При натисканні кнопки Export (Рисунок 5.6), усі дані завантажуються з додатку у файл, котрий потім зберігається у відповідну папку.

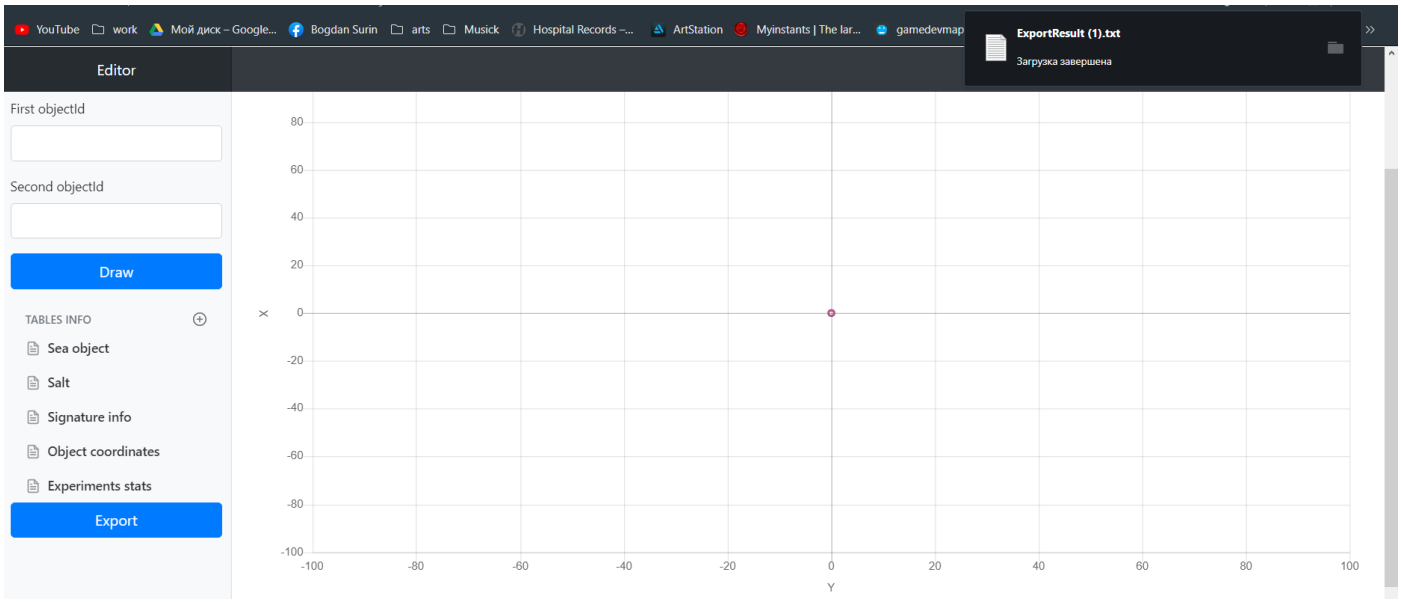

Рисунок 5.6-експорт усіх даних

Вся інформація, експортована з бази даних зберігається у файлі ExportResult (Рисунок 5.7)

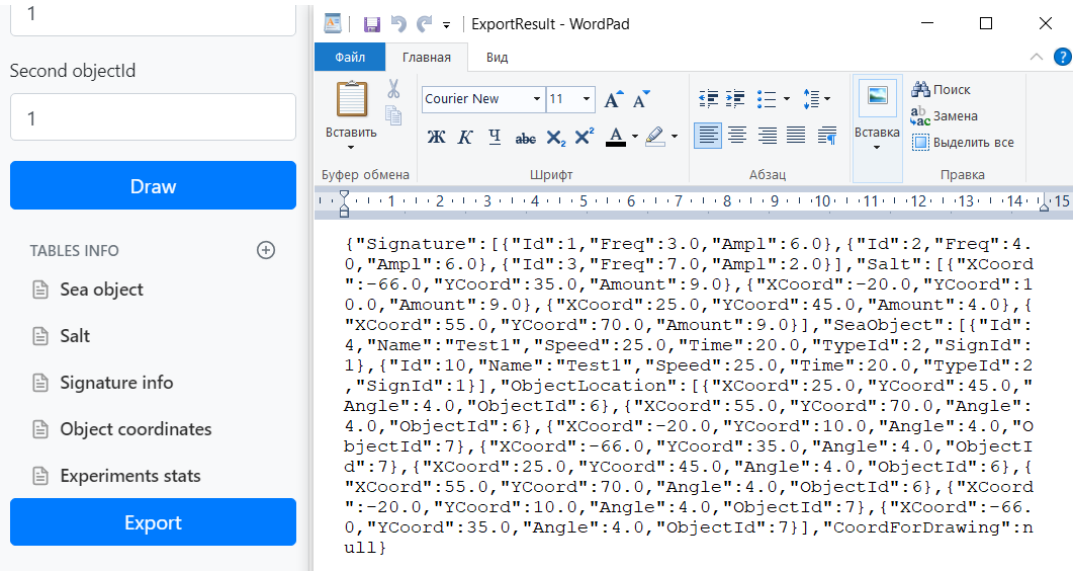

Рисунок 5.7- файл ExportResult

Даний файл у розширенні json зберігає у собі початкові дані експерименту, інформацію з усіх таблиць бази даних, а саме: опис об’єктів, їх координати, швидкість та час, сигнатура та профілі солоності.

ВИСНОВКИ

У ході виконання дипломної роботи було спроектовано модель даних інформаційної системи для предметної області «Веб-редактор сценарію гідроакустичного експерименту». Інформаційне забезпечення включає необхідні засоби для відображення існуючих записів та вибірок даних з таблиць.
Під час виконання дипломної роботи було виконано наступні завдання:

Проведений аналіз існуючих систем гідроакустичного експерименту, та аналогічних веб-додатків для моделювання умов експерименту.

Розроблена інформаційна система включає в себе: базу даних, що містить такі дані, як: інформація про стан об’єкту, його швидкість, тип та місцеположення у просторі, стан солоності а також рівні сигнатур. Структурно програмне забезпечення містить сторінки на локальному сервері, які мають доступ до бази даних.

Програмне забезпечення реалізоване в середовищі Visual Studio, за допомогою мов програмування C# та формальної непроцедурної мови програмування SQL. Для запуску програмного модуля необхідно створити базу даних в середовищі MS SQL, завантажити її та відкрити початкову сторінку у веб-додатку.
Представлений додаток виконує такі функції:
— візуалізація даних на комп’ютері користувача;
— експорт таблиць з бази даних у форматі *.json;
— можливість перегляду завантажених даних;
— запис, редагування та видалення даних у БД;

Під час виконання роботи був отриманий практичний досвід проектування та розробки бази даних, використання моделі даних та їх організації. Також отримано навички створення програмного забезпечення з ефективним та зрозумілим користувацьким інтерфейсом засобами середовища розробки Visual Studio за допомогою однієї мови програмування C#.

Результатом виконання дипломного проекту стала розробка програмного застосунку для задання початкових умов гідроакустичного експерименту та їх експорту в подальші сценарії. Було проведено дослідження існуючих програмних комплексів для гідроакустичного експерименту. В ході роботи було поглиблено знання та навички використання об'єктно-орієнтованої мови та створення гнучкого веб-додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Боббер Р.Дж. Гидроакустические измерения М.: Мир, 1974. – 360 с.

2. Евтютов А.П., Митько В.Б. Инженерные расчеты в гидроакустике 1981 г— 64 с.

3. Ильичев В.И. и др Статистическая теория обнаружения гидроакустических сигналов М.: Наука, 1992. — 415 с.

4. Зарайский В.А. Акустика океана Учебн. пособие - СПб.: Изд-во ВМА им. Кузнецова, 2003. - 233 с.

5. Корякин Ю.А. и др. Корабельная гидроакустическая техника: состояние и актуальные проблемы СПб.: Наука, 2004. — 410 с.

6. Карабанов И.В., Миронов А.С. Алгоритмы обработки гидроакустических сигналов Хабаровск: ТОГУ, 2018. — 140 с.

7. Клюкин И.И. Звук и море Л. «Судостроение», 1974 г. -240с

8. Новиков Б.К. и др. Нелинейная гидроакустика Л.: Судостроение, 1981. — 264 с.

9. Пятакович В.А., Василенко А.М., Хотинский О.В. Распознавание и классификация источников формирования полей различной физической природы в морской среде — Владивосток: Морской государственный университет, 2017. — 255 с.

10. Румынская И.А. Основы гидроакустики — 1979. — 209 с.

11. К. Дж. Дейт SQL и реляционная теория. Как грамотно писать код на SQL. СПб.: Символ-Плюс, 2010. – 480 с.

12. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.

13. Итан Браун Веб-разработка с применением Node и Express. - М. СПб: Издательский дом "Питер", 2016. - 336 с.

14. Кен Хендерсон "Microsoft SQL Server: структура и реализация. Профессиональное руководство" —117 с.

15. Николас Закас ECMASCRIPT 6 ДЛЯ РАЗРАБОТЧИКОВ. Питер, 2017. - 352 с.

16. Облик и структура гидроакустической системы

[Електронний ресурс] — Режим доступу: http://elib.rshu.ru/files_books/pdf/rid_54d82304c38d49699435d8e704980b18.pdf

17. Документація HtmlBook [Електронний ресурс] – Режим доступу до ресурсу: http://htmlbook.ru/

18. Документація Bootstrap-4 [Електронний ресурс] – Режим доступу до ресурсу: https://bootstrap-4.ru/docs/4.5/layout/overview/

19. Документація CSS [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/css/css_grid_layout/auto-placement_in_css_grid_layout

20.Решение уравнений дальности обнаружения в системе гидроакустических расчетов [Електронний ресурс] — Режим доступу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/84834/11 - Samolyuk.pdf?sequence=1

ДОДАТОК А

Web-редактор сценарію гідроакустичного експерименту

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61

Аркушів 9

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 81-1	Сурін_Б. С_ТР61.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-1	Startup.cs	Модулі серверної частини
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-2	HomeController.cs	Модуль-контролер
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-3	AddLocationObject.cs	Модуль додавання координат об’єктів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-4	AddSeaObject.cs	Модуль додавання опису об’єктів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-5	DeleteLocationObjectCommand.cs	Модуль видалення координат об’єктів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-6	DeleteSeaObjectCommand.cs	Модуль видалення опису об’єктів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 13-2	Опис.docx	Опис модуля інтерфейсу клієнтської частини програми

```
namespace Diplom {
   public class Startup    {
      public Startup(IConfiguration configuration)        {
         Configuration = configuration;
      }
      public IConfiguration Configuration { get; }

      // This method gets called by the runtime. Use this method to add services to the container.
      public void ConfigureServices(IServiceCollection services)
      {
         //Connection string(строка для подключения к базе в классе Startup)
         var connectionString = "Data Source = DESKTOP-S9V8MA5\\SQLEXPRESS; Initial Catalog = Edit; Integrated Security = True; Connection Timeout
= 360;";
         services.Configure<CookiePolicyOptions>(options =>
         {
            // This lambda determines whether user consent for non-essential cookies is needed for a given request.
            options.CheckConsentNeeded = context => true;
            options.MinimumSameSitePolicy = SameSiteMode.None;
         });

         //внедрение зависимостей(dependency injection)
         services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
         services.AddSingleton<IGetSignatureInfo>(x =>
          new GetSignatureInfo(connectionString));
         services.AddSingleton<IGetSaltQuery>(x =>
         new GetSaltQuery(connectionString));
         services.AddSingleton<IGetObjectLocationQuery>(x =>
          new GetObjectLocationQuery(connectionString));
         services.AddSingleton<IGetSeaObjectsQuery>(x =>
         new GetSeaObjectQuery(connectionString));
         services.AddSingleton<IDeleteSeaObjectCommand>(x =>
         new DeleteSeaObjectCommand(connectionString));
         services.AddSingleton<IGetCoordByObjId>(x =>
        new GetCoordByObjId(connectionString));
         services.AddSingleton<IAddSeaObjects>(x =>
        new AddSeaObject(connectionString));
         services.AddSingleton<IAddLocationObjects>(x =>
         new AddLocationObjects(connectionString));
         services.AddSingleton<IDeleteLocationObjectCommand>(x =>
         new DeleteLocationObjectCommand(connectionString));
         services.AddSingleton<ISignatureService, SignatureService>();
         services.AddSingleton<ISaltService, SaltService>();
         services.AddSingleton<ISeaObjectService, SeaObjectService>();
         services.AddSingleton<IObjectLocationServicecs, ObjectLocationService>();
         services.AddSingleton<IDrawingService, DrawingService>();
               }
      // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
      public void Configure(IApplicationBuilder app, IHostingEnvironment env)
      {
```

```
            if (env.IsDevelopment())            {
                app.UseDeveloperExceptionPage();
            }
            else        {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseCookiePolicy();
            app.UseMvc(routes =>        {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
                routes.MapRoute(
                    name: "table",
                    template: "{controller=Home}/{action=GetSignInfo}");
            });
        }
    }
}
```



```
using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using Diplom.Models;
using Diplom.Contracts.Domain.Services;
using System.Text;
namespace Diplom.Controllers{
    public class HomeController : Controller    {
        private readonly ISignatureService _signatureService;
        private readonly ISaltService _saltService;
        private readonly ISeaObjectService _seaObjectService;
        private readonly IObjectLocationServicecs _objectLocationServicecs;
        private readonly IDrawingService _drawingService;
            public HomeController(ISignatureService signatureService, ISaltService saltService, ISeaObjectService seaObjectService, IObjectLocationServicecs
objectLocationServicecs,
            IDrawingService drawingService)       {
            _signatureService = signatureService;
            _saltService = saltService;
            _seaObjectService = seaObjectService;
            _objectLocationServicecs = objectLocationServicecs;
            _drawingService = drawingService;
        }
        public IActionResult Index()        {
            return View();
        }
        public IActionResult GetSignInfo()        {
            var result = _signatureService.GetSignature();
            ViewBag.Message = "Sign";
            var commonModel = new CommonViewModel            {
                Signature = result
```

```
    };
    return View("Index", commonModel);
}
public IActionResult GetSaltInfo()      {
    var result = _saltService.GetSalts();
    ViewBag.Message = "Salt";
    var commonModel = new CommonViewModel          {
        Salt = result
    };
    return View("Index", commonModel);
}
public IActionResult GetSeaObjectInfo()      {
    var result = _seaObjectService.GetSeaObjects();
    ViewBag.Message = "SeaObject";
    var commonModel = new CommonViewModel          {
        SeaObject = result
    };
    return View("Index", commonModel);
}
public IActionResult GetObjectLocationInfo()      {
    var result = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "ObjectLocation";
    var commonModel = new CommonViewModel
    {
        ObjectLocation = result
    };
    return View("Index", commonModel);
}
public IActionResult Export()      {
    var signatures = _signatureService.GetSignature();
    var salt = _saltService.GetSalts();
    var seaObject = _seaObjectService.GetSeaObjects();
    var objectLocation = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "ObjectLocation";
    var commonModel = new CommonViewModel          {
        ObjectLocation = objectLocation,
        Salt = salt,
        SeaObject = seaObject,
        Signature = signatures
    };
    var jsonObj = Newtonsoft.Json.JsonConvert.SerializeObject(commonModel);
    string contentType = "application/json";
    string fileName = "ExportResult.json";
    var content = Encoding.ASCII.GetBytes(jsonObj);
    return File(content, contentType, fileName);
    //return View("Index", commonModel);
}
[HttpGet]
public IActionResult DeleteSeaObject()      {
    var result = _seaObjectService.GetSeaObjects();
    ViewBag.Message = "DeleteSeaObject";
    var commonModel = new CommonViewModel          {
        SeaObject = result
```

```
    };
    return View("Index", commonModel);
}

[HttpPost]
public IActionResult DeleteSeaObject(int deleteId)        {
    _seaObjectService.DeleteSeaObject(deleteId);
    var result = _seaObjectService.GetSeaObjects();
    ViewBag.Message = "SeaObject";
    var commonModel = new CommonViewModel            {
        SeaObject = result
    };
    return View("Index", commonModel);
}

[HttpGet]
public IActionResult DeleteLocationObject()        {
    var result = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "DeleteLocationObject";
    var commonModel = new CommonViewModel            {
        ObjectLocation = result
    };
    return View("Index", commonModel);
}

[HttpPost]
public IActionResult DeleteLocationObject(int xCorr, int yCorr)        {
    _objectLocationServicecs.DeleteObjectLocations(xCorr, yCorr);
    var result = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "ObjectLocation";
    var commonModel = new CommonViewModel            {
        ObjectLocation = result
    };
    return View("Index", commonModel);
}

[HttpGet]
public IActionResult AddLocationObject()        {
    var result = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "AddLocationObject";
    var commonModel = new CommonViewModel            {
        ObjectLocation = result
    };
    return View("Index", commonModel);
}

[HttpPost]
public IActionResult AddLocationObject(int XCoord, int YCoord, decimal angle, int objectId)
{           _objectLocationServicecs.AddObjectLocations(XCoord, YCoord, angle, objectId);
    var result = _objectLocationServicecs.GetObjectLocations();
    ViewBag.Message = "ObjectLocation";
    var commonModel = new CommonViewModel            {
        ObjectLocation = result
```

```
            };
            return View("Index", commonModel);
        }
        [HttpGet]
        public IActionResult InsertSeaObject()       {
            var result = _seaObjectService.GetSeaObjects();
            ViewBag.Message = "InsertSeaObject";
            var commonModel = new CommonViewModel           {
                SeaObject = result
            };
            return View("Index", commonModel);
        }
        [HttpPost]
        public IActionResult InsertSeaObject(string name, decimal speed, decimal time, int typeId, int signId)       {
            _seaObjectService.AddSeaObject(name, speed, time, typeId, signId);
            var result = _seaObjectService.GetSeaObjects();
            ViewBag.Message = "SeaObject";
            var commonModel = new CommonViewModel           {
                SeaObject = result
            };
            return View("Index", commonModel);
        }

        [HttpPost]
        public IActionResult DrawGraphs(int firstId, int secondId)       {
            var coordResult = _drawingService.GetCoordById(firstId, secondId);
            ViewBag.Message = "Draw";
            var commonModel = new CommonViewModel           {
                CoordForDrawing = coordResult
            };
            return View("Index", commonModel);
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()       {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

```
using Dapper;
using Diplom.Contracts.DataAccess.Commands;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
namespace Diplom.DataAccess.Commands
{
   public class AddLocationObjects : IAddLocationObjects
   {
      private readonly string _conString;
      public AddLocationObjects(string conString)
      {
         _conString = conString;
      }
      public void Execute(int XCoord, int YCoord, decimal angle, int objectId)
      {
         var input = new DataTable();
         var param = new DynamicParameters();
         param.Add("XCoord", XCoord, DbType.Int32, ParameterDirection.Input);
         param.Add("YCoord", YCoord, DbType.Int32, ParameterDirection.Input);
         param.Add("Angle", angle, DbType.Decimal, ParameterDirection.Input);
         param.Add("ObjectId", objectId, DbType.Int32, ParameterDirection.Input);
         using (var connection = new SqlConnection(_conString))
         {
            try
            {
               connection.Execute
               (
                  "AddLocationObjects",
                  param: param,
                  commandType: CommandType.StoredProcedure,
                  commandTimeout: 0
               );
            }
            catch
            {
               throw;
            }
         }
      }
   }
}
```

```
amespace Diplom.DataAccess.Commands
{
   public class AddSeaObject : IAddSeaObjects
   {
      private readonly string _conString;

      public AddSeaObject(string conString)
      {
         _conString = conString;
      }

      public void Execute(string name, decimal speed, decimal time, int typeId, int signId)
      {
         var input = new DataTable();

         var param = new DynamicParameters();
         param.Add("Name", name, DbType.String, ParameterDirection.Input);
         param.Add("Speed", speed, DbType.Decimal, ParameterDirection.Input);
         param.Add("Time", time, DbType.Decimal, ParameterDirection.Input);
         param.Add("TypeId", typeId, DbType.Int32, ParameterDirection.Input);
         param.Add("SignId", signId, DbType.Int32, ParameterDirection.Input);

         using (var connection = new SqlConnection(_conString))
         {
            try
            {
               connection.Execute
               (
                  "AddSeaObjects",
                  param: param,
                  commandType: CommandType.StoredProcedure,
                  commandTimeout: 0
               );
            }
            catch
            {
               throw;
            }
         }
      }
   }
}
```

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-5

```
using Dapper;
using Diplom.Contracts.DataAccess.Commands;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;

namespace Diplom.DataAccess.Commands{
   public class DeleteLocationObjectCommand : IDeleteLocationObjectCommand    {
      private readonly string _conString;
      public DeleteLocationObjectCommand(string conString)       {
         _conString = conString;
      }
      public void Execute(int XCoord, int YCoord)       {
         var param = new DynamicParameters();
         param.Add("XCoord", XCoord, DbType.Int32, ParameterDirection.Input);
         param.Add("YCoord", YCoord, DbType.Int32, ParameterDirection.Input);
         using (var connection = new SqlConnection(_conString))          {
            connection.Execute              (
               "LocationObjectDelete",
               param: param,
               commandType: CommandType.StoredProcedure,
               commandTimeout: 0
            );
         }
      }
   }
}
```

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 12-6

```
using Dapper;
using Diplom.Contracts.DataAccess.Commands;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
namespace Diplom.DataAccess.Commands{
   public class DeleteSeaObjectCommand : IDeleteSeaObjectCommand    {
      private readonly string _conString;
      public DeleteSeaObjectCommand(string conString)       {
         _conString = conString;
      }
      public void Execute(int id)       {
         var param = new DynamicParameters();
         param.Add("Id", id, DbType.Int32, ParameterDirection.Input);
         using (var connection = new SqlConnection(_conString))          {
            connection.Execute              (
               "SeaObjectDelete",
               param: param,
               commandType: CommandType.StoredProcedure,
               commandTimeout: 0
            );
         }
      }
   }
}
```

ДОДАТОК Б

Web-редактор сценарію гідроакустичного експерименту

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61_61120 20Б 13-2

Аркушів 7

Київ – 2020

АНОТАЦІЯ

Додаток надає можливість моделювання гідроакустичного експерименту та зберігати дані у БД з подальшим експортом.

Спроектоване програмне забезпечення візуалізує морські об’єкти у вигляді точок на уявній мапі та записує інформацію щодо стану об’єктів у таблиці БД з можливістю експорту даних у форматі xml.

Web-сервіс було розроблено за допомогою платформи Microsoft Visual Studio 2019 з використанням мови C# для клієнтської сторони програми.

ЗМІСТ

1. Загальні відомості 4

2. Функціональне призначення 5

3. Опис логічної структури 6

4. Використовувані технічні засоби 7

5. Вхідні і вихідні дані 8

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до теми дипломної роботи, програма має назву «Моделювання гідроакустичного експерименту».

Програма працює на персональному комп’ютері в автономному режимі та не потребує доступу до мережі інтернет, але потребує встановлення на ПК додаткового ПО, що забезпечує швидкість, практичність та надійність.

Система була написана мовою C#, платформи .NET Framework.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб повинен вирішити задачу задання початкових умов експерименту для подальшої обробки тощо. Це було реалізовано за допомогою кількох функцій системи, а саме:

– візуалізація морських об’єктів;

– запис інформації щодо об’єктів та стану водойми до БД;

– експорт даних з БД у вигляді файлу з розширенням xml;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається з клієнтської та серверної частини.

Загальний принцип роботи додатку такий:

1) користувач вводить початкові дані;

2) Натискає на кнопку “Демонстрація”

3) метод, що викликається в обробнику виконує запит;

4) результати виконання методу виводяться на сторінку.

Спочатку проходить запуск форми з усіма її елементами: кнопки, текстові поля, таблиці та частина водного простору. Після того як користувач ввів необхідну інформацію та натиснув на кнопку “ Демонстрація ”, котра знаходиться у формі, викликається метод modelling. В нього передаються усі початкові параметри. Він надсилає запит до сервера та передає до нього всі дані. На сервері проводяться розрахунки розташування об’єктів. Після успішного завершення він повертає результат на клієнтську сторону у вигляді даних місцезнаходження морських об’єктів для подальшої візуалізації. Сама інформація зберігається у БД.

Після цього користувач може експортувати всю необхідну інформацію у вигляді xml файлу через додаткове меню. Воно розташоване у вікні з таблицями БД, яке відкриється натиснувши кнопку “Дані експерименту”.

Після закінчення роботи з цими даними, користувач перезавантажує сторінку для того щоб очистити всі введені початкові дані та розпочинає побудову нового експерименту.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для організації доступу до програмного продукту потрібно мати комп’ютер або ноутбук з необхідним програмним забезпеченням.

Кінцевим користувачам для роботи з програмою потрібно, щоб на комп’ютері був встановлений Microsoft SQL, а операційна система була Windows 7,8 або 10.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

—координати морського об’єкта;

—швидкість морського об’єкта;

—кут відносно початку координат

— глибина водного середовища;

— початковий час;

— кінцевий час;

— частота коливань;

— амплітуда коливань;

Вихідними даними є:

— візуалізація об’єктів експерименту;

— згенерований xml-файл з усією інформацією;